COMMODORE

PER UTENTI DI C64 - C16 - PLUS-4 LIRE 5000

Linguaggio Macchina dall'A alla Z

Il disassemblato delle routine grafiche di Toma

Febbraio 1987 - Anno IV - N° 17 - Sped. Abb. Post. Gr. III/70 - Distr. MePe.

SPECIALE DA COLLEZIONE

Systems

COMMODORE

DIRETTORE
Alessandro de Simone

SEGRETERIA DI REDAZIONE:
Maura Ceccaroli

UFFICIO GRAFICO:
Arturo Ciaglia

EDIZIONI:
Systems Editoriale S.r.l.
(Registro Nazionale Stampa n. 01500

DIREZIONE, REDAZIONE, PUBBLICITA'
Viale Famagosta, 75 - 20142 Milano
Tel. 02/8467348 - Autorizzazione del Tribunale di Milano N. 103 del 25/2/84
Direttore responsabile:
Agostina Ronchetti

COMPOSIZIONI:
Systems Editoriale S.r.l.

FOTOLITO:
Systems Editoriale S.r.l.

STAMPA:
La Litografica - Busto Arsizio (VA)
Rotostampa - Brugherio

Concessionario esclusivo per la diffusione MePe Spa
Via G. Carcano, 32 - Milano

COMMODORE

speciale

Con questo numero speciale ci rivolgiamo ai lettori, vecchi e nuovi, di riviste specifiche per i prodotti Commodore.

Ci auguriamo che i lettori ne decretino il successo, incoraggiandoci a proseguire nella pubblicazione di una collana dedicata a monografie di particolare interesse.

Il lettore più attento avrà notato con piacere che gli argomenti trattati nel fascicolo sono due e non uno soltanto, come abbiamo appena affermato.

I due argomenti, però, (Linguaggio Macchina e Grafica del C/64) sono slegati tra loro soltanto in apparenza. Come ognuno ben sa, infatti, è impossibile gestire la grafica in alta risoluzione limitandosi al linguaggio Basic.

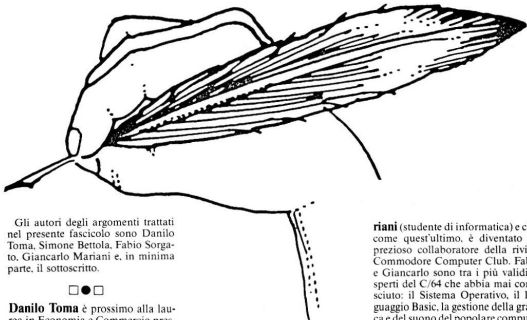
Risulta indispensabile affidarsi interamente al Linguaggio Macchina (o, quantomeno, all'Assembly) per generare animazioni, disegni e grafici ad una velocità soddisfacente.

Ecco, dunque, giustificata pienamente la scelta di informare il lettore, grazie ad una parte propedeutica sul linguaggio macchina, indispensabile per comprendere pienamente la seconda parte, specifica sulle routine grafiche.

Sia che il nostro lettore sia interessato allo studio del solo linguaggio macchina, sia che desideri approfondire la conoscenza della gestione della grafica, il presente fascicolo rappresenterà un prezioso manuale da tenere sempre a disposizione nei "pressi" della tastiera.

La comprensione degli argomenti trattati non sarà privilegio di pochi, come, purtroppo, capita di costatare esaminando pubblicazioni analoghe: numerosi programmi di applicazione, da provare con gran facilità, esempi semplici e alla portata di tutti (anche dei principianti), schemi di immediata comprensione, non potranno che facilitare l'ingresso del lettore nel meraviglioso mondo dei microprocessori e della gestione della memoria.

Alessandro de Simone

Gli autori

Gli autori degli argomenti trattati nel presente fascicolo sono Danilo Toma, Simone Bettola, Fabio Sorgato, Giancarlo Mariani e, in minima parte, il sottoscritto.

□●□

Danilo Toma è prossimo alla laurea in Economia e Commercio presso l'università Bocconi di Milano. Riservato, studioso e oltremodo simpatico, si è dedicato allo studio del Sistema Operativo del Commodore 64 fin da quando riuscì a procurarsene uno dei primi esemplari (agli inizi dell'83). La carenza di software di quei tempi lo costrinse a realizzare numerosi programmi di utility che lo aiutassero nella programmazione in Linguaggio Macchina e che propose, per la pubblicazione, alla Systems Editoriale. La prima stesura di un programma che consentiva lo sfruttamento intensivo del linguaggio macchina per la gestione della grafica venne pubblicato sulla rivista Commodore Computer Club N.10 (aprile '84). La versione che, invece, consentiva di aggiungere nuovi comandi Basic specifici per la grafica tridimensionale, vide la luce poco più tardi (C.C.C. N.14). L'entusiasmo dimostrato dai lettori per le ormai famose Routine Grafiche di Toma lo indusse a ampliare il set dei comandi fino a giungere alla versione definitiva, pubblicata nel presente fascicolo.

Simone Bettola, neo matricola del Politecnico di Milano, ha iniziato per caso la collaborazione con la Systems Editoriale il giorno in cui capitò in Redazione per la richiesta di alcuni numeri arretrati. Bastò una breve conversazione per capire che l'allora studente del Liceo Scientifico aveva stoffa per trattare argomenti impegnativi. Gli fu affidato il compito di proseguire le puntate sul linguaggio macchina, appena iniziato su Commodore Computer Club, e di collaborare con altre iniziative software della Systems Editoriale. Verrà tra breve impegnato nello sviluppo del settore Hardware e nell'affrontare tematiche relative a sistemi superiori, come l'Ms-Dos. Attualmente sviluppa sofware per simulazioni di carattere scientifico e non disdegna di lavorare in Assembler sul PC IBM.

Fabio Sorgato è uno studente del liceo scientifico che ho conosciuto grazie al suo amico **Giancarlo Mariani** (studente di informatica) e che, come quest'ultimo, è diventato un prezioso collaboratore della rivista Commodore Computer Club. Fabio e Giancarlo sono tra i più validi esperti del C/64 che abbia mai conosciuto: il Sistema Operativo, il linguaggio Basic, la gestione della grafica e del suono del popolare computer non hanno segreti per loro che, nei ritagli di tempo, si dilettano a scovare i "bug" del C/64. Tra le loro opere più recenti, a parte i numerosi programmi e articoli pubblicati sulle riviste, segnaliamo l'ultimo nato, il simulatore di Gw-Basic, scritto l'estate scorsa rinunciando, in parte, alle vacanze. Fabio e Giancarlo, attualmente, oltre che a collaborare con la Systems Editoriale, sono anche consulenti di informatica presso software house. Sono, entrambi, poco più che diciottenni.

Alessandro de Simone, cioè il sottoscritto, dirige la rivista "Commodore Computer Club" e la sezione "Commodore" della rivista "Personal Computer". Non volendo, per ovvi motivi, tracciare un autoritratto, preferisco rispondere, a chi mi chiede di che cosa mi occupo, che faccio il Talent Scout: di giovani, naturalmente, "condannati" a diventare affermati professionisti di domani...

Alessandro de Simone

La Grande Libreria Systems

Autori Vari

64 Programmi per Commodore 64

Giochi, grafica, gestione delle stringhe, musica, numeri, gestionali.

Lire 4.800

Autori Vari

I miei amici C16 & Plus4

Un manuale pratico per padroneggiare il basic di questi computer.

Lire 7.000

Autori Vari

Strategie vincenti per Commodore 64

Le strategie per tutti i classici del videogioco: per giocarli, vincerli o programmarli.

Lire 5.800

Autori Vari

62 Programmi per il Vic 20, C16 e Plus 4

Giochi, grafica e routine per imparare a programmare.

6.500

Roberto Didoni, Guido Grassi

Utilities e giochi didattici

Raccolta di programmi pratici per tutti i Commodore e lo Spectrum.

Lire 6.500

Giovanni Mellina

Tutti i segreti dello Spectrum

4 passi nella Rom: come usare le più importanti routine del sistema operativo.

Lire 7.000

Roberto Didoni, Guido Grassi

Simulazioni e test per la didattica

Teoria e listati per Vic 20, C16, C64 C128 e Spectrum Sinclair.

Lire 7.000

Paolo Goglio

Impara giocando il basic dello Spectrum

Esercizi pratici per entrare nel vivo della programmazione.

Lire 7.000

Clizio Merli

µPascal per Commodore 64/128

Un manuale completo per il programma compilatore

Lire 7.000

Umberto Colapicchioni e Luca Galuzzi

Dal registratore al drive del C64

Tutti i segreti delle memorie di massa del Commodore 64

Lire 7.000

Autori Vari

ADA

Il linguaggio passepartout dei computer degli anni '80.

Lire 5.000

Clizio Merli

Il linguaggio PASCAL

Un manuale tascabile per lo studio e la programmazione.

Lire 5.000

Sì, voglio arricchire la mia biblioteca con i seguenti volumi al prezzo di copertina + lire 3.000 per spese di spedizione.

- ☐ **64 Programmi per Commodore 64**
- ☐ **Strategie vincenti per i tuoi videogames**
- ☐ **62 Programmi per Vic 20 C16 e Plus77**
- ☐ **Utilities e giochi didattici**
- ☐ **Tutti i segreti dello Spectrum**
- ☐ **Simulazioni e test per la didattica**
- ☐ **Imparare giocando il basic dello Spectrum**
- ☐ **I miei amici C16 e Plus4**
- ☐ **Pascal per Commodore 128**
- ☐ **Dal registrattore al drive del C64**
- ☐ **ADA**
- ☐ **Il linguaggio Pascal**

Nome ....

via .... **N.ro.** .... **telefono** ....

CAP .... **Città** ....

Su tale importo mi praticherete lo sconto del 10% in quanto abbonato a ☐ *Commodore Computer Club* ☐ *Personal Computer* ☐ *Com puter* ☐ *VR Videoregistrare*. Pertanto vi invio la somma soltanto di lire ....

Valore dell'ordine lire....................

Ritagliare e spedire in busta chiusa regolarmente affrancata a Systems Editoriale - V.le Famagosta, 75 - 20142 Milano.

"Commodore Speciale" tutto su disco!

I lettori interessati a... non digitare i (lunghi) programmi pubblicati su questo fascicolo, possono far richiesta del dischetto contenente TUTTI i listati Basic di "Commodore Speciale". Sono presenti anche numerosi altri programmi, già pubblicati su C.C.C. che saranno utilissimi grazie alla stretta attinenza con gli argomenti trattati nel presente fascicolo.

Per essere più espliciti, pubblichiamo qui di seguito la Directory del floppy disk che può esser vostro per una cifra addirittura ridicola.

Ricordiamo che i programmi possono esser richiesti SOLTANTO su dischetto: si prega di astenersi dal chiederli su nastro, onde evitare spiacevoli rifiuti.

Il costo da sostenere per la riproduzione dei nastri, infatti, ci costringerebbe ad offrire il package a prezzi non in linea con la nostra politica commerciale impegnata ad offrire software di qualità a cifre irrisorie.

Coloro che desiderano procurarsi il package su disco devono infatti limitarsi ad inviare in Redazione la modica cifra di L.12000, (oltre a L.3000 per le spese di spedizione). Non ci è possibile, infatti, inviare materiale contrassegno.

Compilate un normale modulo di C/C postale indirizzando a:

C/C postale N. 37952207
Systems Editoriale
Viale Famagosta, 75
20142 Milano

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento") non solo il vostro nominativo completo, ma anche il nome del software desiderato: "Commodore Speciale: le routine grafiche su disco."

Chi volesse ricevere più celermente la confezione può inviare la somma richiesta mediante assegno circolare oppure normale assegno bancario non trasferibile (o barrato due volte) intestato a: Systems Editoriale, Milano.

Un normale modulo di conto corrente, infatti, impiega circa due mesi (provare per credere!) per giungere al nostro indirizzo. Una lettera, invece, arriva anche in una sola settimana.

Inviate l'assegno, accompagnato da una lettera in cui specificate la richiesta, al solito indirizzo:

Systems Editoriale
Viale Famagosta, 75
20142 Milano

```
0    "COMMODORE SPEC. " CS 2A
1    "----PARTE 1-----"  PRG
2    "0 PROGRAMMA"       PRG
2    "1 PROGRAMMA"       PRG
2    "2 PROGRAMMA"       PRG
3    "3 PROGRAMMA"       PRG
3    "4 PROGRAMMA"       PRG
3    "5 PROGRAMMA"       PRG
3    "6 PROGRAMMA"       PRG
3    "7 PROGRAMMA"       PRG
3    "8 PROGRAMMA"       PRG
3    "9 PROGRAMMA"       PRG
3    "10 PROGRAMMA"      PRG
3    "11 PROGRAMMA"      PRG
4    "12 PROGRAMMA"      PRG
4    "13 PROGRAMMA"      PRG
17   "DISASSEMBLER 5"    PRG
32   "IPER MONITOR LM"   PRG
16   "ASSEMBLER"         PRG
1    "----PARTE 2-----"  PRG
37   "ROUTINE GRAF/1"    PRG
2    "DISAB.RESTORE"     PRG
7    "LOAD/SAVE PAG.G."  PRG
9    "CHAR+INV(20000)"   PRG
9    "LENTE 25000-"      PRG
4    "ASSEMBLATORE R."   PRG
4    "H-COPY MPS/802"    PRG
1    "DEMO ARTE ASTRAT"  PRG
1    "DEMO INV"          PRG
2    "DEMO CHAR/VCHAR"   PRG
2    "DEMO LENS"         PRG
5    "DEMO P.TO VISTA"   PRG
1    "DEMO RAGGIERA"     PRG
2    "DEMO POLIG/STELL"  PRG
2    "DEMO SCRITTA CBM"  PRG
2    "DEMO COSENO 3D"    PRG
2    "DEMO MULTICOLOR"   PRG
1    "DEMO SPIRALE"      PRG
2    "DEMO FUNZIONE 3D"  PRG
3    "DEMO MOVIMENTO"    PRG
4    "DEMO CUBO"         PRG
3    "DEMO PAESAGGIO"    PRG
1    "DEMO CHIOCCIOLA"   PRG
1    "DEMO CIELO STELL"  PRG
2    "DEMO FARFALLA 1"   PRG
2    "DEMO FARFALLA 2"   PRG
1    "DEMO RAGNATELA"    PRG
1    "DEMO SPIRALI"      PRG
1    "DEMO TERZA DIMEN"  PRG
2    "DEMO QUADRATO 3D"  PRG
4    "DEMO GRAFICI"      PRG
1    "DEMO FIORE"        PRG
6    "H-COPY MPS/803"    PRG
1    "----------------"  PRG
4    "GRAVITAZ.PLOTTER"  PRG
3    "GRAVITAZ. VIDEO"   PRG
13   "ASTA CHE CADE"     PRG
28   "ISOMERI C-64"      PRG
12   "ISTOGRAMMI HI-RE"  PRG
19   "DIGITALIZZATORE "  PRG
19   "64 SIMULA ROBOT"   PRG
5    "AUTOMA CELLULARE"  PRG
10   "FIGURE GEOMETR."   PRG
5    "RICORSIVITA' C64"  PRG
19   "GIOCO"             PRG
291 BLOCKS FREE.
```

Corso di linguaggio macchina

Premessa

Questo mini-corso è destinato a fornire una certa conoscenza della programmazione in Linguaggio Macchina a chi non si è mai avvicinato alla struttura interna del proprio computer e che avverte ormai le limitazioni imposte dal Basic.

Innanzitutto è bene chiarire alcuni concetti di base: il Linguaggio Macchina ed il linguaggio Assembler (che non sono sinonimi, anche se in definitiva possono essere considerati uno come la semplificazione dell'altro) non sono riservati ad una ristretta elite di geniacci computeromani con almeno tre lauree.

Anzi, potrà sembrare strano, ma, parlando in termini di Informatica, il linguaggio più semplice in assoluto (per la macchina, naturalmente) viene considerato appunto il Linguaggio Macchina; a questo fa seguito l'Assembler, che (come vedremo) grazie alla tecnica dei codici mnemonici, si presta maggiormente ad essere utilizzato dall'uomo, ferma restando la stretta corrispondenza con il precedente.

Il Basic, come il Pascal, il Cobol, il Forth ed altri ancora, sono linguaggi più "evoluti", perchè più vicini all'uomo, ma risultano di difficile interpretazione per la macchina. All'ultimo gradino dei linguaggi vi sono i cosiddetti "applicativi", che riguardano quasi esclusivamente i Personal Computer veri e propri che si occupano più da vicino di un particolare settore di interesse, come la gestione di testi, gli archivi, i Data Base e via dicendo; si tratta di veri e propri linguaggi di particolare semplicità e praticità per l'utente, ma che nascondono programmi di dimensioni iperboliche.

Conosciamo la CPU

Ma non allontaniamoci troppo e torniamo al nostro microprocessore. Questo egregio componente, che può essere chiamato anche "chip" dagli elettronici oppure "CPU" (Central Process Unit), dagli informatici, è il cuore di tutto il sistema. Nel caso del Commodore 64, del Vic 20, del C16, del Plus4 e del C128 (nella versione 64) il microprocessore è uguale a parte le sigle: si tratta del famigerato 6510 (6502 e 7501). Per comodità faremo riferimento a questo, poichè tra i vari chip variano solo alcune caratteristiche elettroniche che non interessano in questa sede.

La peculiarità del chip è quella di eseguire una serie di operazioni riconoscendole e differenziandole tra loro grazie ad un codice numerico in base binaria di otto cifre. Che cosa vuol dire tutto ciò? Semplicemente che, a seconda del codice, il circuito integrato è in grado di compiere operazioni elementari sui registri interni e sulla memoria del computer. Conoscere il Linguaggio Macchina significa conoscerne i codici e sapere quale particolare operazione è abbinata a ciascuno di essi.

L'inizio

Quando si parla con i possessori di calcolatrici programmabili che intendono allargare le proprie conoscenze sui computer, ci si sente chiedere quante linee-programma e quanti dati può memorizzare un personal, evidenziando, appunto, una delle differenze notevoli dell'organizzazione dati-programmi esistenti tra i personal e le programmabili.

In effetti è più corretto dire che il microprocessore ed il Sistema Operativo del computer, o della calcolatrice, sono gli elementi che gestiscono i dati in un modo per cui, a livello macroscopico, è possibile parlare di dati, programmi, istruzioni, eccetera.

A livello elementare, invece, un microprocessore, qualunque esso sia, tratta in modo automatico il contenuto di una successione di locazioni di memoria che di per sè possono contenere, indifferentemente, dati oppure istruzioni.

Un microprocessore, pertanto, lavora in un modo che è quello conforme alle specifiche della casa costruttrice. Chi costruisce un computer, che si basa su un particolare processore, deve crearvi "attorno" il cosiddetto "Firmware" rappresentato dal Sistema Operativo (O.S.) ed, eventualmente, dal linguaggio che sarà adoperato sul computer. In una macchina da calcolo si ha in definitiva:

- HARDWARE, rappresentante l'insieme di connessioni elettriche, circuiti stampati, integrati, tastiera, video, periferiche, le cui caratteristiche l'utente difficilmente potrà modificare.

- FIRMWARE: è l'insieme dei circuiti integrati (I.C. oppure C.I.) in genere

a sola lettura (ROM, EPROM, PROM) contenenti l'O.S. ed il linguaggio, quando c'è.

Il firmware si può definire come l'organizzazione che gestisce le funzionalità del microprocessore. Esso, differentemente dall'hardware, può essere modificato facilmente dall'utente. E' noto che in alcuni sistemi, sostituendo le ROM, è possibile far parlare il computer non più in Basic, ma, ad esempio, in Pascal.

Altri esempi di firmware sono rappresentati da integrati venduti da alcune ditte, da inserire negli zoccoli vuoti per I.C. presenti sul circuito stampato, per aumentare le capacità di calcolo del computer (calcolo matriciale, word processor, eccetera).

• SOFTWARE: è quello interamente gestibile dall'utente ed è rappresentato dalle RAM che sono particolari I.C. in cui è possibile sia scrivere che leggere informazioni. Quando, ad esempio, facciamo girare un programma scritto in Basic, in effetti è il programma LM scritto su firmware che interpreta ciò che abbiamo digitato sulla tastiera o introdotto da disco piuttosto che da nastro, ed impone al microprocessore l'elaborazione dei dati tali da soddisfare le richieste dell'utente. Ecco perchè quando si sceglie un computer, può essere utile sapere il "peso" del linguaggio adoperato e del Sistema Operativo. Grosso modo un Basic da 8K è di certo meno sofisticato di uno da 16K, ma più di uno da 4K. Anche il "peso" dell'O.S. può dare un'idea della complessità, e quindi della capacità di elaborazione, di un Sistema Operativo.

Ciò premesso ci occuperemo della organizzazione e di alcune istruzioni che un microprocessore, ed esattamente il 6502, è capace di eseguire.

Che cosa è un numero?

Prima di procedere è opportuno soffermarci sul concetto di "numero" dal momento che abbiamo anticipato alcuni concetti di base. Dicevamo prima, che il codice riconosciuto dal computer è un insieme di otto cifre in base 2, vale a dire che gli unici valori emessi e accettati dal microprocessore sono lo zero e l'uno, che vengono chiamati anche livelli logici perchè corrispondono ad una particolare situazione elettronica di presenza (1) oppure di assenza (0) di tensione. Questi otto stati o livelli logici, combinandosi diversamente tra di loro, possono dare luogo a ben 256 combinazioni differenti.

Per quanto riguarda il computer, è necessario chiarire che ogni singolo valore binario (zero o uno) viene chiamato "bit", mentre un insieme di otto bit viene definito "byte" o "parola". Nel nostro caso specifico, il 6502 è un microprocessore ad 8 bit. Ciò significa appunto che questo è in grado di trattare simultaneamente fino ad otto cifre binarie, e che una sua "parola" è composta da otto bit: ogni locazione di memoria può contenere un numero compreso tra zero e 255.

Ciò è giustificato dal fatto che un microprocessore ad otto bit (come, appunto, quelli montati sui computer Commodore), possono "trattare" otto simboli per volta che, ciascuno, possono valere solo "1" oppure "0".

Vediamo di chiarire questo importante concetto, che rappresenta il fondamento dell'informatica.

Noi siamo abituati già alla notazione binaria (=due simboli soltanto) anche senza che ce ne rendiamo conto: Consideriamo, infatti, gli indicatori lampeggianti di direzione di una qualsiasi autovettura ed esaminiamo i vari casi che possono presentarsi:

• Nessuna delle due lampadine lampeggia. Ciò significa che l'auto procede diritta (oppure è parcheggiata!) e l'automobilista non ha intenzione di svoltare.

• Lampeggia il solo indicatore destro. Ciò significa che tra breve l'automobile girerà a destra.

• Lampeggia l'indicatore sinistro. L'auto svolterà a sinistra.

• Lampeggiano entrambe le lampadine. L'automobile è ferma in corsia di emergenza.

Badate bene che con sole due lampadine (destra e sinistra) è possibile generare quattro significati diversi.

Il numero quattro deriva dal numero due (le possibilità, cioè lo stato di "acceso" oppure "spento") elevato al quadrato (il numero delle lampadine).

Schematizzando l'esempio, le quattro possibilità possono esser indicate nel modo seguente:

Lampadine

Sinis.	Des.
Ø	Ø
Ø	1
1	Ø
1	1

Nel caso di un semaforo, che è formato da tre lampade, sarebbe possibile generare un numero di due (stati possibili) elevato alla terza (numero di lampade), cioè otto simboli diversi. Schematizzando come prima, si ha:

Lampade

Verde	Giallo	Rosso	
0	0	0	(Semaforo non funz.)
0	0	1	(Alt)
0	1	0	(Attenzione)
0	1	1	
1	0	0	(Via libera)
1	0	1	
1	1	0	(Affrettarsi)
1	1	1	

Come si può notare, si possono individuare, nel traffico stradale, un numero inferiore di stati possibili.

Se invece di lampadine parliamo di bit e, invece di acceso e spento, parliamo di "uni" e di "zeri", il discorso di base sul Linguaggio Macchina non costituisce più alcun problema.

Dunque, riepilogando: Avendo a disposizione otto bit (lampadine) ognuno dei quali può essere in uno solo di due stati (1, 0 oppure acceso o spento, oppure ON, OFF eccetera), quante possibili combinazioni è possibile ottenere?:

2 elevato all'ottava potenza = 256 simboli diversi, proprio come abbiamo detto prima.

Il "primo" è costituito da un gruppo di otto zeri (00000000), l'ultimo da un insieme di otto "uni" (11111111).

Il calcolatore, dunque, "ragiona" solo con gli zeri e gli uni (che brutti termini!). Sarebbe un guaio se anche noi dovessimo scrivere un programma ricorrendo a due soli simboli. Fortunatamente c'è un sistema che

consente di trasformare un qualsiasi numero binario (formato normalmente da otto bit) nel corrispondente decimale. Esempio: A quale numero decimale corrisponde il numero binario di otto bit 10010101?

Si considerano i simboli ad uno ad uno. Questi rappresentano l'"esistenza" (se è eguale ad uno) della potenza di due corrispondente. In caso contrario (=0) ne denotano l'assenza. Spieghiamoci meglio scrivendo il numero binario in verticale anziché in orizzontale. Vale a dire, invece di 10010101, scriviamolo:

1	La	corrispondente	potenza	di	due	ESISTE
Ø	"	"	"	"	"	NON "
Ø	"	"	"	"	"	NON "
1	"	"	"	"	"	ESISTE
Ø	"	"	"	"	"	NON "
1	"	"	"	"	"	ESISTE
Ø	"	"	"	"	"	NON "
1	"	"	"	"	"	ESISTE

Che cosa, però, intendiamo con il termine "corrispondente"?

Semplicemente il "posto" che quel simbolo occupa nella numerazione all'interno del byte (insieme costituito dagli otto bit). Il primo simbolo a sinistra è il settimo, il secondo è il sesto e così via, fino al primo a destra che rappresenta il bit di "peso" zero.

Peso (pos.)	Val. (Ø/1)	Esiste (si/no)	Calcolo (pot.2)	Risult.
7	1	si	2↑7	128
6	Ø	no	Ø	Ø
5	Ø	no	Ø	Ø
4	1	si	2↑4	16
3	Ø	no	Ø	Ø
2	1	si	2↑2	4
1	Ø	no	Ø	Ø
Ø	1	si	2↑Ø	1

Si ricorda che il numero due (e qualsiasi altro numero) elevato alla potenza zero fornisce come risultato "1" e non zero come potrebbe sembrare a prima vista.

Si noti, inoltre, che il "peso", cioè le posizioni, sono numerate da "7" a "0" e non da "8" a "1": anche zero è un numero!

Il numero binario 10010101 corrisponde, dunque, al decimale: 128+16+4+1=149

La notazione binaria è quindi costituita da due soli simboli (0/1); quella binaria da dieci (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). E quella esadecimale, di cui si sente spesso parlare? Da ben sedici simboli. Questi sono i "soliti" dieci della decimale, più le prime sei lettere dell'alfabeto A, B, C, D, E, F.

Per ciò che riguarda la corrispondenza tra le due notazioni, si tenga presente che per i valori tra zero e nove, la decimale e la esadecimale sono perfettamente identiche. Le cose cambiano da 10 a 15. Vediamo:

Decim	Esad.
9	9
1Ø	A
11	B
12	C
13	D
14	E
15	F

E per i numeri maggiori di 15? E' intuitivo...

Decim	Esad.
16	1Ø
17	11
...	...
31	1F
...	...
255	FF

Per esercizio, "inventatevi" un numero compreso tra zero e 255 e "traducetelo" in binario e in esadecimale. In seguito utilizzate il programma riportato in queste pagine per controllarne la corretta conversione.

```
 90 REM PROGRAMMA Ø
 95 :
100 REM CONVERSIONE
110 REM DA DECIMALE
120 REM A BINARIO
130 REM ED ESADECIMALE
140 :
150 INPUT "NUMERO DECIMALE (Ø-2
    55)";ND
160 X=ND:A$=""
170 IF ND>255 OR ND<Ø OR ND<>IN
    T(ND) THEN 15Ø
180 FOR I=Ø TO 7:Y=X/2
190 IF Y<>INT(Y) THEN A$="1"+A$
    :GOTO 21Ø
200 A$="Ø"+A$
210 X=INT(Y):NEXT:PRINT"BINARIO
     "A$
220 X=ND/16:Y=INT(X):X=(X-Y)*16
230 PRINT"ESADECIMALE ";
240 IF Y<1Ø THEN PRINTY;:GOTO 2
    6Ø
250 PRINTCHR$(55+Y);
260 IF X<1Ø THEN PRINTX;:PRINT:
     RUN
270 PRINTCHR$(55+X):PRINT: RUN
```

Esistono anche computer più evoluti, in grado di trattare simultaneamente fino a 16 o anche fino a 32 bit. Ma la notevole velocità è pagata dalla maggiore complessità delle istruzioni.

Tutto sommato, quindi, la soluzione degli 8 bit resta forse una delle più valide per l'utilizzo non professionale, anche se non proprio dilettantesco.

Continuiamo ora il discorso sul Linguaggio Macchina, trattando la corrispondenza esistente tra numerazione decimale e binaria. Per esempio, per noi il numero 183 significa un numero di oggetti pari alla somma di un centinaio, otto decine e tre unità. Volendo esprimere questo numero come un insieme di potenze di dieci, noi scriviamo:

183=
1*10↑ 2+
8*10↑ 1+
3*10↑ 0

Si ricordi che qualsiasi numero, tranne zero, elevato a potenza nulla, fornisce come risultato il numero 1 e non zero, come abbiamo già sottolineato. Inoltre si ricordi che 10↑ n è u-

guale a 10 * 10 * 10 * ... 10 n volte. Per esempio:
10 ↑ 4= 10*10*10*10= 10000.

Qualsiasi numero, pertanto, può venir rappresentato come somma di potenze di dieci, decrescenti (2,1,0, nel nostro caso, perchè 183 è composto di tre cifre) ciascuna moltiplicata per un fattore che è una delle cifre del numero considerato. Tale sistema di numerazione è stato adottato dall'uomo perchè, probabilmente, quando scoprì i numeri si servì del metodo più semplice di cui potesse disporre: le dita delle mani.

Un calcolatore, e quindi anche un Commodore, ha a disposizione soltanto uno stato alto ed uno basso, a seconda se in un particolare punto del circuito vi è tensione o meno. Indicando lo stato alto di tensione con "1" e quello basso con "0", il computer dispone di un sistema di numerazione che ha appena due simboli, e prende appunto il nome di Sistema Binario. Una cifra, in tale sistema, prende il nome di bit ed una quantità qualunque deve venire espressa come potenza di due.

Supponiamo di avere in ingresso otto segnali corrispondenti ad una situazione elettronica di questo tipo:

0 1 0 1 1 0 1 1

Questo valore è espresso in codice binario, esattamente allo stesso modo in cui noi scriviamo i consueti numeri che sono in base 10. Il "nostro" 123 equivale in pratica a:

1*10 ↑ 2+
2*10 ↑ 1+
3*10 ↑ 0=
123

Allo stesso modo avremo che quell'accozzaglia di uno e di zero che abbiamo visto prima corrisponde ad uno e ad un numero soltanto, determinabile matematicamente. Avremo:

0*2 ↑ 7+
1*2 ↑ 6+
0*2 ↑ 5+
1*2 ↑ 4+
1*2 ↑ 3+
0*2 ↑ 2+
1*2 ↑ 1+
1*2 ↑ 0=
91

Il problema si pone quando si tratta di memorizzare quantità decisamente grandi, anche se di uso frequente (dell'ordine del milione), per le quali c'è bisogno di decine di simboli binari contro i sette o al massimo otto del decimale.

Non è difficile accorgersi della notevole complessità di un tale sistema di numerazione, almeno per l'uomo, che è abituato a ragionare in base 10 (chissà se avessimo avuto solo 8 dita cosa sarebbe successo!).

Per ovviare a questo inconveniente, dato che, al contrario, la macchina si trova a proprio agio, i matematici hanno pensato bene di dividere quella cifra così complessa in due parti (nibble) attribuendo a ciascuna di esse un codice alfanumerico a seconda del numero indicato.

Così %0101 = $5 e %1011 = $B, dove il simbolo "%" indica un numero in notazione binaria, mentre il simbolo "$" indica la numerazione esadecimale (a base 16); in questa, oltre ai numeri, sono utilizzate anche le prime sei lettere dell'alfabeto e ciò per sopperire alla mancanza di adeguati simboli per indicare numeri superiori a 10. Questo fatto, anzichè complicare la situazione, come potrebbe sembrare a prima vista, la semplifica, perchè accostando i due numeri ottenuti ($5 e $B) otteniamo la cifra esadecimale $5B che corrisponde appunto al nostro #91 calcolato prima ("#" è il simbolo della base 10):

5*16 ↑ 1+11*16 ↑ 0=91

I vantaggi della numerazione esadecimale sono molteplici: inanzitutto qualsiasi valore riconoscibile dal microprocessore sarà espresso con due soli simboli, dal momento che con 8 valori binari si può ottenere, oltre allo zero, il valore massimo di 255 (decimale) corrispondete ad $FF esadecimale. In secondo luogo, ma non meno importante, così facendo il programmatore ha sempre sott'occhio la situazione interna (e quindi binaria) della macchina, perchè con un po' di allenamento non è difficile imparare a memoria la tabellina di conversione da binario (con 4 sole cifre) ad esadecimale (fino ad "F"), per poi fare rapidamente tutte le conversioni desiderate. Queste due caratteristiche, da sole, hanno fatto sì che il Linguaggio Macchina assumesse come caratteristica principale l'uso di codici esadecimali, sottintendendo l'origine binaria di tale scelta.

Si può dunque immaginare facilmente che una certa quantità sarà rappresentata da un numero di simboli esadecimali, inferiore a quello decimale e a sua volta nettamente inferiore al binario puro.

Poichè la memoria di un calcolatore è una successione di gruppi di otto bit ciascuno, noi potremo considerare, per semplicità, ciascun byte come se fosse formato da due valori esadecimali, benchè, nella realtà, il valore sia "scritto" in binario puro. Si presenta ora un altro problema nel progetto di un calcolatore, e cioè come individuare, tra le tante, una certa locazione di memoria. Ad ogni byte si assegna un indirizzo che altro non è se non un gruppo di sedici bit.

In tale modo si possono indirizzare 2 ↑ 16 = 65536 locazioni di memoria. Con un numero inferiore di bit (ad esempio 8), si possono indirizzare solamente 2 ↑ 8 = 256 locazioni di memoria, insufficienti per un versatile uso di un computer. Da notare che, in generale, i microprocessori ad otto bit (6502, 8080, Z80, ecc.) trattano come dati gruppi di otto bit, ma hanno come "indirizzi" gruppi di 16 bit.

Per poter gestire una memoria più grande, dato che in teoria non c'è relazione tra numero di bit di un dato e numero di bit di un indirizzo, si potrebbero avere dati di otto bit, ma indirizzi di trentadue o comunque più di sedici bit.

Come mai, quindi, non si sceglie questa soluzione per aumentare la capacità di memoria, e invece si ricorre a memorie di massa, come i nastri magnetici, i floppy disk o alle tecniche più sofisticate, come le memorie virtuali?

Semplicemente perchè sarebbe, in un primo caso, più complesso, come vedremo in seguito, indirizzare oltre il valore di 65536; in secondo luogo, risulterebbe necessario cambiare l'architettura stessa della CPU, e di conseguenza, modificare la struttura dei linguaggi già diffusi sul mercato internazionale.

Conversioni tra diverse basi numeriche

Ritornando agli indirizzi e ai dati, vediamo ora di individuare un indirizzo di cui sappiamo il valore solo in decimale o esadecimale.

Il problema si presenta quando si usano, da Basic, i comandi PEEK e POKE, dato che siamo costretti a fornire gli argomenti dei comandi stessi espressi come valori decimali, mentre spesso li conosciamo come valori esadecimali.

Poniamo per esempio di voler convertire il numero 11052 decimale nel corrispondente esadecimale (che ha sedici simboli), ma il ragionamento che seguiamo sarà simile per qualunque sistema di numerazione.

Si divide il numero in oggetto per il numero di simboli del sistema scelto e si considera il resto ed il quoziente intero (primo resto = 12; primo quoziente = 690). Il quoziente, se maggiore o uguale a sedici, si divide nuovamente per sedici (secondo resto = 2; secondo quoziente = 43): poichè il nuovo quoziente è ancora maggiore di sedici, si ripete il procedimento finchè si ottiene un quoziente minore di sedici (terzo resto = 11; terzo quoziente = 2).

Il numero esadecimale desiderato è formato dai tre resti ottenuti e dall'ultimo quoziente in ordine inverso: 2, 11, 2, 12.

Infine, sostituendo tali valori con i simboli corrispondenti in esadecimale, si ottiene il valore cercato: 2B2C.

Convertiamo ora un numero esadecimale in decimale, limitandoci a trattare sòlamente i numeri compresi tra 0000 e FFFF. La prima cifra, delle quattro, rappresenta il numero moltiplicato per 16 3; la seconda per 16 2, poi 16 1 e quindi 16 0: pertanto, volendo convertire 2B2C si scrive:

2*16↑3+
B*16↑2+
2*16↑1+
C*16↑0

convertendo i numeri "B" e "C" esadecimali in decimale otteniamo:

2*16↑3+
11*16↑2+
2*16↑1+
12*16↑0=
11052

Usiamo PEEK e POKE

Prima di continuare è opportuno saper usare correttamente le istruzioni PEEK e POKE del Basic.

Queste istruzioni consentono di leggere (PEEK) in tutta la memoria e di scrivere (POKE) un qualsiasi numero intero, compreso tra 0 e 255, in qualsiasi locazione della RAM. Per esempio, se noi battiamo:

PRINT PEEK(4080)

apparirà, in decimale, il valore della 4080ma locazione RAM. Viceversa, battendo:

POKE 4080,151

il valore 151 decimale sarà trascritto nella 4080ma locazione di memoria.

Da notare, però, che mentre il comando PEEK(X) si limita a leggere un valore e non lo modifica in nessun caso, l'istruzione POKE X,Y cerca di modificare il valore della locazione X. Possono, quindi, verificarsi alcuni casi critici.

• Cerchiamo di scrivere in una locazione ROM del Basic oppure dell'O.S. Naturalmente, il dato che cerchiamo di scrivere non viene scritto, in quanto nella ROM (Read Only Memory, memoria a sola lettura) non si può scrivere, e nessun messaggio di errore appare per informarci dell'impossibilità di eseguire l'operazione.

• Analoga mancanza di messaggio di errore si verifica se l'indirizzo della POKE cade in una zona RAM (Random Access Memory, memoria ad accesso casuale) non esistente nella configurazione del sistema, ma questa evenienza è impossibile da verificarsi con il Commodore 64 dato che ad ogni indirizzo corrisponde un "oggetto".

• L'indirizzo della POKE rappresenta una locazione RAM utilizzata dall'O.S. e una sua modifica può "distruggere" (non irrimediabilmente per la macchina, comunque) il sistema. Saremo, in questo caso, costretti a spegnere e poi a riaccendere il computer, perdendo, purtroppo, tutto il contenuto della RAM.

• Il valore Y di POKE X,Y è negativo o maggiore di 255. Questo è l'unico caso in cui compare un messaggio di errore. Teniamo presente, infine, che se Y non è un valore intero, verrà presa in considerazione solo la parte intera.

Supponiamo di non cadere in uno dei casi critici: come facciamo allora a memorizzare un numero maggiore di 255? Il procedimento è un po' lungo, ma è l'unico che sia possibile adottare.

Spacchiamo un byte

Vogliamo memorizzare il numero intero 11052 decimale. Abbiamo già visto che in esadecimale corrisponde a 2B2C, e poichè ogni cifra esadecimale rappresenta quattro bit, in totale avremo bisogno di sedici bit. Dato che ogni locazione di memoria contiene otto bit, noi potremo "spezzare" 2B2C in due gruppi: 2B e 2C; il primo, dopo averlo tradotto in decimale, lo scriveremo in una locazione, il secondo in quella successiva.

Riassumiamo il procedimento in altre parole:

11052 decimale = 2B2C esadecimale.
Separiamo 2B da 2C

2B esa. = 2*16↑1 + 11*16↑0 = 43 dec.
2C esa. = 2*16↑1 + 12*16↑0 = 44 dec.

Utilizziamo, per le verifiche che seguono, le locazioni RAM a partire da 49152 decimale fino 53247 (esa C000 / CFFF), perchè tali locazioni sono destinate a programmi creati dall'utente e quindi non si rischia di "distruggere" il sistema.

Per tranquillizzare il lettore preciseremo ciò che si intende per "distruzione" del sistema.

L'errore più "grave" che si può commettere è il tentativo di scrivere un dato in una zona ROM invece che

RAM. Anche se questa operazione viene eseguita milioni di volte (e può capitare se si incorre in un loop chiuso) nessun danno può esser provocato alla ROM nè a qualsiasi altro circuito elettronico del computer usato.

Se un programma L.M. è scritto male, oppure mal strutturato, può capitare che i numerosi tentativi effettuati per "riprendere" il controllo del computer (Run/Stop e Restore, tasto di Reset eccetera) non sortiscano alcun effetto. In casi come questi l'unico modo per riprendere il lavoro consiste nello spegnere e riaccendere l'apparecchio (e le periferiche ad esso collegate) perdendo, però, il programma digitato.

Ecco perchè è buona norma registrare SEMPRE un programma L.M. oppure Assembler PRIMA di mandarlo in funzione. Se non dovesse funzionare, non sarete costretti a riscriverlo ex-novo, ma sarà sufficiente ricaricarlo (da nastro o disco) ed esaminarlo attentamente in modo da rintracciare (ed eliminare) l'errore.

Riprendiamo il discorso interrotto scrivendo:

POKE 49152,44: POKE 49153,43

Come si può notare, del numero 2B2C abbiamo trascritto nella prima locazione (49152) il valore 2C (LSB: Least Significant Byte = byte meno significativo) e nella successiva (49153) il 2B (MSB: Most Significant Byte = byte più significativo), anzichè al contrario, come saremmo stati indotti a fare istintivamente.

Adottiamo questo procedimento dato che lo segue anche la CPU. Come faremo allora a sapere quale numero è rappresentato da due locazioni di memoria successive? Naturalmente seguendo il ragionamento inverso.

Traduciamo il MSB in decimale e lo moltiplichiamo per 256:
2B esadecimale = 43 decimale; 43 * 256 = 11008

Analogamente ci comportiamo per il LSB, moltiplicandolo, però, per 1 (lasciandolo cioè invariato):
2C esadecimale = 44 decimale

In seguito eseguiamo la somma tra i due:
11008 + 44 = 11052

I numeri negativi

Come si può facilmente verificare, il numero più grande che si può rappresentare è 65535 ed il più piccolo è zero, ma comunque, tutti positivi. Se però rinunciamo ad ottenere valori così grandi, possiamo seguire la convenzione secondo cui sono da considerare positivi i valori fino al numero (65535-1)/2, cioè da zero a 32767, mentre da 32768 a 65535 inclusi sono negativi e valgono esattamente il valore considerato meno 32767.

Per esempio, il numero 52768 corrisponderà con questa convenzione a:
52768 = -(52768-32767) = -20001

In questo modo possiamo rappresentare tutti i numeri interi negativi e positivi compresi fra -32768 e +32767. Ecco spiegato, dunque, perchè certi computer non molto sofisticati (ed anche i Commodore quando si usano variabili intere del tipo A%) trattano solamente quei numeri interi: con due byte adiacenti non è possibile superare l'intervallo suddetto.

Ricapitoliamo dunque i concetti espressi finora:

• Il calcolatore ragiona solo in binario puro, tratta cioè solo gruppi di otto stati di tensione elettrica alla volta, alta o bassa, detti BIT.
• Per semplicità il dato formato da otto bit (detto parola o byte) viene spezzato in due da quattro bit (detti ciascuno Nibble) e tradotto in esadecimale, al solo scopo di rendere la vita più semplice al programmatore.
• Per semplificare ancor più la stesura di un programma in Linguaggio Macchina si ricorre spesso ad un linguaggio detto Assembler che, utilizzando gruppi di lettere derivati dalle iniziali delle parole inglesi che indicano la funzione dell'istruzione, ed incolonnati uno dopo l'altro, consente una relativa facilità nell'individuare eventuali errori o nell'apportare modifiche.

Assembler: chi era costui?

Come si vedrà meglio in seguito, nel linguaggio Assembler non è necessario ricordare a memoria tutte le istruzioni del 6502 sotto forma di coppie di valori esadecimali, nè tantomeno indicare volta per volta certe locazioni con l'indirizzo in esadecimale. Dobbiamo ricordare, comunque, che ancora più che nel caso del Basic, non esiste un Assembler universale, ma vi sono in commercio diversi tipi di Assemblatori che differiscono l'uno dall'altro in alcuni particolari.

Tali programmi permettono di compilare (tradurre da linguaggio mnemonico in Linguaggio Macchina) in Assembler, ma non sono assolutamente indispensabili per seguire i programmi presentati via via in questo corso.

Potrà comunque essere utilissimo quale compendio per i lettori che vorranno stendere programmi per conto proprio o per scrivere quei programmi pubblicati da altre riviste (eventualità rara, molto rara...) che non riportano anche il cosiddetto codice oggetto, cioè il programma scritto direttamente in Linguaggio Macchina.

Innanzitutto è opportuno conoscere il sistema secondo il quale è possibile programmare il nostro computer. Per quanto riguarda il Linguaggio Macchina vero e proprio, l'unico metodo di programmazione consiste nell'inserire in memoria (con delle POKE) i codici e gli operandi, rispettando un preciso ordine stabilito precedentemente su di un supporto... cartaceo. Assai più comodo risulta essere l'impiego di speciali programmi assemblatori e disassemblatori, di cui parlavamo prima, denominati "MONITOR" (Machine Language Monitor, per esteso), che evitano il fastidio di convertire i codici mnemonici in numeri e di pokarli in memoria.

Il primo programma!

L'aspetto più curioso, se vogliamo, del discorso, è il fatto che dopo aver inserito in memoria dei semplici numeri, il microprocessore distingua tra di loro quali corrispondono ad istruzioni e quali sono invece operandi, da considerarsi esclusivamente come numeri. Consideriamo il seguente esempio:

LDA #$00

```
60130 DATA 078,201,200,176,235
60140 DATA 165,078,041,007,133
60150 DATA 247,165,078,074,074
60160 DATA 041,254,168,185,235
60170 DATA 193,056,229,247,133
60180 DATA 247,185,236,193,024
60190 DATA 101,088,133,248,165
60200 DATA 087,041,248,168,165
60210 DATA 087,041,007,170,024
60220 DATA 096,032,036,159,032
60230 DATA 094,197,173,236,159
60240 DATA 208,008,189,227,193
60250 DATA 049,247,056,208,001
60260 DATA 024,008,032,109,197
60270 DATA 040,096,032,003,159
60280 DATA 169,008,032,071,171
60290 DATA 169,199,133,251,169
60300 DATA 000,133,087,133,088
60310 DATA 133,079,162,007,134
60320 DATA 253,165,251,133,078
60330 DATA 032,111,159,102,252
60340 DATA 198,078,198,253,208
60350 DATA 245,102,252,165,252
60360 DATA 009,128,032,071,171
60370 DATA 165,145,201,127,240
60380 DATA 032,230,087,208,002
60390 DATA 230,088,165,088,240
60400 DATA 212,165,087,201,064
60410 DATA 208,206,169,013,032
60420 DATA 071,171,165,251,056
60430 DATA 233,007,133,251,201
60440 DATA 252,208,182,169,015
60450 DATA 032,210,255,169,013
60460 DATA 032,210,255,076,028
60470 DATA 159,000
```

Mappa della memoria

Configurazione della memoria da $C000 a $CFFF dopo il caricamento delle routine grafiche, della modifica per la disattivazione del tasto Restore, dei comandi per il caricamento e registrazione della pagina grafica, di CHAR/UCHAR e di INV..

Indirizzi	Contenuto
C000 – C2D1	ROUT. GRAF. ORIGINALI
C2D2 – C2DF	DISABIL. TASTO RESTORE
C2EC – C3FB	ROUT. GRAF. ORIGINALI
C3FD – C41B	TABELLA INDIRIZZI
C41C – C426	ULTERIORE SPAZIO PER INDIRIZZI
C427 – C7AB	ROUT. GRAF. ORIGINALI
C7AC – C7FE	TABELLA DELLE PAROLE
C7FF – C8B3	SPAZIO DISPONIBILE PER PAROLE O PER I DATI DEGLI SPRITE
C8B4 – C969	ROUT. GRLOAD/GRSAVE/ECC.
C970 – CA52	ROUT. CHAR/UCHAR
CA53 – CA72	ROUT. INV
CA73 – CBFF	SPAZIO DISPONIBILE PER SPRITE
CC00 – CFE7	MEMORIA DI SCHERMO HIRES
CFF8 – CFFF	PUNTATORI DEGLI SPRITES HIRES

STA $0400
RTS

Convertendolo in codici macchina otteniamo:

A9 00
8D 00 04
60

Se volessimo rendere esecutivo il programma (il cui scopo è quello di far comparire una chiocciolina in alto a sinistra sullo schermo), dovremmo innanzitutto decidere in quale zona di memoria vogliamo allocare i dati rappresentanti il programma LM.

```
 10 REM  PROGRAMMA 1
 20 PRINTCHR$(147)
 30 PRINT"STAMPA UNA CHIOCCIOLI
    NA"
 40 PRINT"IN ALTO A SINISTRA"
 50 PRINT"SULLO SCHERMO."
100 READ A$:IF VAL(A$)<0 THEN P
    RINT"ATTIVARE CON: SYS49152
    ":END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(
    RIGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
     140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
     160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=A
    D+1:GOTO 100
200 DATA A9,00,8D,00,04,60,-1
```

CHIOCCIOLINA SULLO SCHERMO

IND.N	ESA	DATI	MNEMON.
49152	C000	A9 00	LDAIM 0
49154	C002	8D 00 04	STA 1024
49157	C005	60	RTS

Questo concetto è molto importante, ed è alla base di tutto il discorso. In Linguaggio Macchina non è il computer (come nel Basic) ad assegnare automaticamente l'area di memoria che ritiene più opportuna, ma siamo noi che dobbiamo decidere, in base alle nostre esigenze, dove "sistemare" il programma.

In fase sperimentale abbiamo già visto che è consigliabile utilizzare la memoria RAM compresa fra 49152 ($C000) e 53247 ($CFFF) decimali. Questa zona, di 4 Kbyte, è molto comoda, a patto che non si usi un Monitor che risieda in quelle locazioni. Comunque sia, qualsiasi parte di memoria è programmabile dall'utente (tranne ovviamente la ROM).

Un programma Basic che consenta di scrivere le nostre routine in linguaggio macchina si può presentare, ad esempio, col seguente aspetto:

```
 10 REM  PROGRAMMA 2
 11 :
 20 PRINTCHR$(147)
 30 PRINT"STAMPA UN CARATTERE"
 40 PRINT"IN UN PUNTO QUALSIASI
    "
 50 PRINT"SULLO SCHERMO."
100 READ A$:IF VAL(A$)<0 THEN P
    RINT"ATTIVARE CON: SYS49152
    ":END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(
    RIGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
     140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
     160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=A
    D+1:GOTO 100
170 REM A0=CARATTERE: A5,04=LOC
    AZIONE VIDEO
200 DATA A9,A1,8D,A5,04,60,-1
```

CARATTERE SULLO SCHERMO

IND.N	ESA	DATI	MNEMON.
49152	C000	A9 A0	LDAIM 160
49154	C002	8D A5 04	STA 1189
49157	C005	60	RTS

Una volta eseguito il nostro programma "caricatore", la routine è pronta per essere utilizzata. A questo punto non ci resta che digitare RUN e, quando compare il Ready, SYS 49152: il nostro primo programma in Linguaggio Macchina verrà eseguito!

Prima ancora di commentare ciò che è successo all'interno del computer, osserviamo che l'indirizzo di partenza specificato dopo la SYS è determinante ai fini dell'esecuzione del programma stesso. E' opportuno sapere che la CPU è microprogrammata (via hardware) in modo tale che riconosca il primo valore che incontra come un'istruzione e non come un operando.

Se per caso dovesse incontrare un codice inesistente nel set a sua disposizione, il sistema si "impallerebbe", rendendo necessario lo spegnimento del computer e la conseguente perdita del programma.

C'è da ricordare, inoltre, che ogni istruzione richiede un diverso numero di operandi: vi sono casi in cui l'istruzione non richiede altre specificazioni (istruzioni ad un byte), casi in cui viene utilizzato un solo operando (istruzione a due bytes) e casi in cui servono due operandi (istruzioni a tre bytes).

Ma prima di procedere in questa direzione, è bene dare un'occhiata più da vicino al sistema in cui vengono gestiti i dati all'interno del computer.

I registri della CPU

La CPU contiene al suo interno alcuni registri (vale a dire: locazioni di memoria "speciali") che consentono sia di effettuare operazioni aritmetiche di vario tipo, sia di rendere possibile il funzionamento dello stesso circuito integrato. Sicuramente non avrebbero alcun significato parole come "programma" e "memoria" se non esistesse un particolare registro chiamato "Program Counter" (contatore di programma, PC per gli amici).

Il PC, come dice il termine stesso, provvede a tenere il conto delle locazioni di memoria nelle quali risiede il programma LM che sta girando. Automaticamente e regolarmente, grazie ad un preciso timer interno, il PC si incrementa a seconda del numero di byte occupati dal codice e dal suo eventuale operando. Nel caso del nostro mini-programmino, dopo il comando Basic SYS 49152 il PC sarà caricato con il valore 49152.

La prima istruzione è A9 00, a due bytes. Quindi, dopo averla eseguita, il PC si autoincrementerà automaticamente di due unità, divenendo uguale a 49154, locazione a partire dalla quale vi è memorizzata, guarda caso, la successiva istruzione, 8D 00 04, a tre bytes.

Questa volta l'incremento sarà uguale a tre unità, ed il PC conterrà il valore 49157, corrispondente all' istruzione 60 (RTS); a questo punto l'esecuzione del sottoprogramma termina, e dal momento che la chiamata era stata effettuata dal Basic, l'interprete stesso riprenderà in mano la situazione.

Il procedimento effettivo che viene svolto all'interno del microprocessore in realtà è molto più complesso, prevedendo precise temporizzazioni che variano da istruzione a istruzione, ma per il momento può essere sufficiente quanto è stato detto.

Non sarà sfuggito ai lettori più attenti il fatto che il PC deve essere in grado di indirizzare 65535 bytes, mentre un registro (che altro non è se non una normale locazione di memoria fisicamente residente nella CPU) può arrivare ad un massimo di 256 (con soli 8 bit...). Anche in questo caso è la matematica che fornisce la soluzione.

In realtà esistono due PC, rispettivamente il Program Counter Low (PCL) e il Program Counter High (PCH). Per ottenere indirizzamenti che giungano fino ai 64 Kbytes di memoria, la CPU provvede a dividere questo numero in due parti che non superino come valore massimo il 255. La formuletta in questione è la seguente: PC = PCL + 256*PCH

Un po' come la faccenda dei nibbles, la comodità è tutta a vantaggio del codice esadecimale, in quanto basta accostare i due valori per ottenere l'indirizzo completo; supponendo che PCL contenga $02 e che PCH sia uguale a $C0, il PC risulterà essere $C002.

Il fatto che abbiamo indicato in tutti questi esempi prima il byte basso (Low Byte o LB) e poi quello alto (High Byte o HB), come abbiamo già avuto modo di osservare, non è stato casuale, anzi! In tutti i casi in cui si richieda un riferimento per qualsiasi locazione di memoria, è questo lo standard adottato dalla CPU, che prevede sempre di seguire rigidamente la successione indicata. Infatti, volendo controllare il banale esempio presentato poco fa, l'istruzione STA $0400 è stata codificata come 8D 00 04, con evidente inversione di posizione dell'operando.

Esula dagli scopi di questo corso fornire la motivazione di questa scelta, che naturalmente non è casuale, ma implica notevoli vantaggi.

Tre registri basilari

Oltre al PC vi sono tre registri singoli, che vengono chiamati "A", "X" e "Y". La loro caratteristica consiste nel fatto che, oltre alla notevole rapidità con la quale vengono gestiti dalla CPU (in media quasi doppia di qualsiasi memoria esterna), esistono particolari istruzioni che consentono di "giocare" con loro. A parte gli scherzi, di questi tre il più importante è di gran lunga il registro A (Accumulatore). E' questo il registro che consente di accedere direttamente alla ALU (Aritmetic Logic Unit), unità di calcolo del microprocessore. E' in questo registro che, prima del calcolo, risiede uno dei due addendi e, dopo l'operazione, il risultato dell'addizione o della sottrazione effettuata (uniche operazioni aritmetiche previste dal 6510).
Per quanto riguarda la comunicazione con l'esterno del microprocessore, vi è il Bus Dati, vero e proprio "canale elettronico" dentro al quale passano i dati sia in ingresso che in uscita.

In seguito ad un'istruzione del tipo LDA $0400, ad esempio, il Bus Dati conterrà innanzitutto la richiesta di accesso alla locazione $0400. La risposta che otterrà sarà immagazzinata nell'accumulatore, pronta per successive elaborazioni.

A questo punto pensiamo sia conveniente fermarsi per meditare un po' su questa notevole massa di concetti. Ma per non lasciare a bocca asciutta chi ci ha seguito sino a questo punto, proponiamo alcuni programmini che vedremo di commentare assieme.

Dati o istruzioni?

Innanzitutto è bene conoscere alcune istruzioni basilari che faciliteranno la comprensione degli stessi. L'istruzione Basic che permette di scrivere un valore, compreso tra zero e 255, direttamente in una locazione RAM, è, come abbiamo già visto, POKE. La stessa istruzione, scritta però in Linguaggio Macchina che utilizza la notazione esadecimale, è 8D. Tale simbolo deve essere seguito, nella memoria del calcolatore, da due byte indicanti rispettivamente la parte bassa (LSB) e la parte alta (MSB) dell'indirizzo in cui si desidera scrivere il contenuto dell'accumulatore

Quanto asserito può lasciare perplesso il lettore: tutti gli statements del linguaggio Basic sono in effetti parole inglesi o parti di esse, di immediata interpretazione. Ci si deve però convincere che un microprocessore "parla" un linguaggio tutto suo, formato esclusivamente da 0 ed 1, e pertanto 8D, rappresentante la traduzione in esa che per noi non significa assolutamente nulla è per esso un preciso comando. In altre parole è solo questione di simboli: ognuno parla la propria lingua.

Un'altra istruzione frequentissima nei programmi in L.M. è A9. Per esempio, A9 A0 viene interpretata dal 6502 nel modo seguente:

"Carica l'accumulatore col contenuto del byte che segue immediatamente l'istruzione stessa (cioè A0)".

Quando il micro avrà eseguito l'istruzione (di due byte, a differenza di 8D, istruzione a tre byte), all'interno dell'accumulatore sarà presente, nel caso specifico, il valore A0. Consideriamo ora il seguente programma:

```
A9 A0          8D A5 04          60
```

Supponiamo di caricarlo in memoria utilizzando un qualsiasi Monitor oppure il programma Basic che proponiamo:

```
10 READ X: IF X = -1 THEN
END
20 POKE 49152+I,X
30 I=I+1
40 GOTO 10
50 DATA 169,160,141,165,004,096,-1
```

Dopo aver digitato SYS 49152, verrà eseguito il programma in L.M. la cui prima istruzione è rappresentata dal contenuto di 49152 ($C000), e cioè A9 (traduzione di 169).

Il 6502 stabilisce, proprio in base all'ordine da noi impartito, che in $C000 vi è un'istruzione e non un dato, ed esegue l'elaborazione del programma partendo da tale unico presupposto. Il micro, da parte sua, sa che A9 è un'istruzione a due byte, la esegue come prima detto e subito dopo esamina il byte-istruzione successivo a quello contenente A0, che è $C002 (in cui è 8D). In quest'ultimo caso l'istruzione è a tre byte. Dopo averla eseguita, il 6502 incontrerà $60, che significa: "Ritorna dalla subroutine". Per motivi che non stiamo qui a

specificare, quando viene incontrata l'istruzione 60 senza che precedentemente sia stata data l'istruzione "Jump to Subroutine", (come nel nostro caso) si ritornerà al Basic.

Che cosa succede impartendo il comando SYS 49152 ?

• A9 A0. Nell'accumulatore viene trascritto il valore A0.

• 8D A5 04. Il valore dell'accumulatore (cioè A0) viene depositato nell'indirizzo $0402 appartenente alla memoria di schermo: apparirà appunto il simbolo corrispondente ad A0.

• 60. Si ritorna al Basic e appare il consueto READY.

Notate che il 6502 stabilisce automaticamente se un'istruzione è a tre, due oppure un solo byte in quanto al suo interno è presente un decodificatore indicante il corretto procedimento da seguire a seconda della istruzione incontrata.

Per insistere su quanto detto, digitiamo ora (sbagliando con intenzione) SYS 49153 (invece di SYS 49152) e cerchiamo di capire che cosa accade:

• A0 8D. Dato che ordiniamo di partire non più da $C000, ma da $C001, il 6502 stabilisce che in tale locazione vi è la prima istruzione da eseguire, il cui significato è completamente diverso da A9 A0 vista nel precedente esempio. Si ricorda che A0 8D significa: carica il registro Y col valore 8D.

• A5 04. Carica nell'accumulatore il valore che ora è presente nella locazione $04 della pagina zero, cioè in $0004.

• 60. Ritorna al Basic.

Benchè non vi sia nulla di errato nella logica del programma così eseguito, non si verifica nulla di particolare, nè tantomeno viene stampato sullo schermo un simbolo come nel primo caso esaminato.

Digitiamo ora SYS 49154 dimostrando nuovamente la notevole importanza del primo indirizzo:

• 8D A5 04. Verrà trasferito, nella locazione $04A5 dello schermo, il contenuto dell'accumulatore presente in quel particolare momento dell'elaborazione. Tale contenuto può essere diverso da quello da noi impostato prima, ed esattamente è quello che il computer si ritrova dopo aver eseguito la routine di interpretazione del comando SYS 49154.

• 60. Si ritorna al Basic come prima.

Anche digitando SYS 49155 non avviene nulla di particolare. Battendo invece SYS 49156, il 6502 cerca di eseguire l'istruzione allocata in $C004, ma poichè $04, a differenza delle precedenti istruzioni, non esiste nel set delle 151 istruzioni del microprocessore, il 6502, molto probabilmente, perde il controllo e non riusciamo più a ripristinare il sistema se non spegnendo e riaccendendo l'apparecchio.

Quest'ultimo caso, come del resto i precedenti, dimostra che è di fondamentale importanza indicare con precisione l'indirizzo di partenza del programma in L.M.

Sbagliando indirizzo si ha, nel più fortunato, e purtroppo raro, dei casi, l'esecuzione di un programma diverso dalle nostre aspettative e, più spesso, la "distruzione" del sistema.

Prime considerazioni

Si potrebbe obiettare che il medesimo risultato ottenuto grazie al programma appena esaminato si poteva ottenere semplicemente mediante il Basic come, ad esempio:

```
10 POKE 1189,160
```

Ciò è vero, ma c'è da fare una prima considerazione: per fare apparire in L.M. un simbolo sullo schermo abbiamo utilizzato solamente 6 byte. Col Basic, invece, occuperemmo ben 10 byte, senza contare che si è dovuto usare l'interprete del Basic impiegando molto tempo. La sostanziale differenza fra i due metodi si nota quando si vorrà riempire tutto lo schermo con il carattere desiderato. Con il Basic scriveremmo, ad esempio:

```
10 A=1
20 B=1000
30 PRINT CHR$(147)
40 FOR I=A TO B
50 PRINT"A"
60 NEXT
70 END
```

Eseguendo il programma si può nettamente seguire il formarsi dei 1000 caratteri "A".

Proviamo ora a caricare il programma in L.M. che segue a partire dalla locazione 826 (=$033A), che è utilizzata solo durante le operazioni di caricamento da cassetta.

```
 10 REM PROGRAMMA 3
 15 :
 20 PRINTCHR$(147)
 30 PRINT"RIEMPIE TUTTO LO SCHE
    RMO CON"
 40 PRINT"IL CARATTERE DESIDERA
    TO"
 50 PRINT"SELEZIONATO TRAMITE:"
 60 PRINT"POKE 847,X":PRINT"ATT
    ENDERE..."
100 READ A$:IF VAL(A$)<0 THEN P
    RINT"ATTIVARE CON: SYS826":
    END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(
    RIGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
     140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
     160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+
    1:GOTO 100
200 DATA A9,04,8D,52,03,A9,00,8
    D
210 DATA 51,03,A9,00,8D,56,03,A
    9
220 DATA D8,8D,57,03,A9,FF,8D,0
    0
230 DATA 07,A9,0C,8D,00,DB,EE,5
    1
240 DATA 03,EE,56,03,D0,EE,EE,5
    2
250 DATA 03,EE,57,03,AD,52,03,C
    9
260 DATA 08,D0,E1,60,-1
```

```
UN CARATTERE SULL'INTERO SCHERMO
--------------------------------

IND.N   ESA     DATI       MNEMON.
--------------------------------

 826   033A   A9 04      LDAIM 4
 828   033C   8D 52 03   STA 850
 831   033F   A9 00      LDAIM 0
 833   0341   8D 51 03   STA 849
 836   0344   A9 00      LDAIM 0
 838   0346   8D 56 03   STA 854
 841   0349   A9 D8      LDAIM 216
 843   034B   8D 57 03   STA 855
 846   034E   A9 FF      LDAIM 255
 848   0350   8D 00 07   STA 1792
 851   0353   A9 0C      LDAIM 12
 853   0355   8D 00 DB   STA 56064
 856   0358   EE 51 03   INC 849
 859   035B   EE 56 03   INC 854
 862   035E   D0 EE      BNE 238
 864   0360   EE 52 03   INC 850
 867   0363   EE 57 03   INC 855
 870   0366   AD 52 03   LDA 850
 873   0369   C9 08      CMPIM 8
 875   036B   D0 E1      BNE 225
 877   036D   60         RTS
```

Esaminiamo, una alla volta, le istruzioni contenute:

PC=$033A: A9 04. Nell'accumulatore viene scritto il numero $04 (in effetti viene scritto il numero binario puro 0000 0100 e non ripeteremo più questo concetto).

PC=$033C: 8D 52 03. Il valore ora presente nell'accumulatore ($04) viene riportato così come è nella locazione $0352. Attenzione che nell'accumulatore è ancora presente $04.

PC=$033F: A9 00. Nell'accumulatore viene ora scritto $00 cancellando il precedente $04; naturalmente ciò che è stato scritto precedentemente in $0352 non viene modificato.

PC=$0341: 8D 51 03. Analoga all'istruzione $033C.

PC=$0344: A9 00. Vedi istruzione $033F.

PC=$0346: 8D 56 03. Vedi istruzione $033C

PC=$0349: A9 D8. Vedi istruzione $033A.

PC=$034B: 8D 57 03. Vedi istruzione $033C

PC=$034E: A9 FF. Vedi istruzione $033A.

PC=$0350: 8D xx xx. I due gruppi di xx stanno a significare che possiamo scrivere qualsiasi valore esadecimale; infatti, quando in seguito faremo partire il programma, le istruzioni locate in $033C e $0341 provvederanno a scrivere la parte alta e bassa dell'indirizzo; a questo punto dell'esecuzione del programma, appare in alto a sinistra la vocale maiuscola "A", che vogliamo trascrivere anche nella seconda posizione, terza, eccetera, fino a riempire tutto lo schermo. Dobbiamo pertanto fare in modo di impartire l'istruzione 8D 01 04 e poi 8D 02 04.., fino a 8D FF 07. Per evitare di scrivere mille istruzioni (quante sono le celle dello schermo) utilizziamo un loop (o iterazione) allo stesso modo del programma Basic.

PC=$0353: A9 0C. Vedi istruzione $033A.

PC=$0355: 8D xx xx. Vedi istruzione $0350 (si occupa della memoria colore allo stesso modo della memoria schermo).

PC=$0358: EE 51 03. Questa istruzione ($EE) aumenta di una unità il contenuto della locazionne il cui indirizzo è $0356: da $00 passa a $01, oppure, quando sarà contenuto A8, passerà ad A9 oppure (importante) da $FF a $00.

PC=$035B: EE 56 03. Vedi istruzione $0358 (si occupa sempre dell'incremento, ma del "puntatore" alla memoria colore).

PC=$035E: D0 EE. L'istruzione $D0 è una istruzione a due byte che fa parte delle cosiddette istruzioni di "BRANCH" (salto condizionato). Arrivati a questo punto, il proseguimento dell'elaborazione dipende dal risultato dell'ultima operazione effettuata.

Nel caso specifico, se EE 56 03 fornisce un valore uguale a $00, l'istruzione $D0 viene ignorata ed il programma prosegue con l'istruzione successiva ($0360). In caso contrario la CPU legge il primo bit del secondo byte dell'istruzione (%1110 1110) e, se esso è 0, eseguirà un salto in avanti, mentre se è 1 (come nel nostro caso) farà un salto indietro.

Per salto si intende il numero di byte (e NON di istruzioni) che bisogna escludere per rintracciare la successiva istruzione da eseguire. Nel nostro caso la CPU esegue un conteggio "alla rovescia" partendo dall'istruzione successiva a $D0 definendola come $FF. In seguito va all'indietro fino a che arriva al numero contenuto nel secondo byte di $D0, cioè $EE, corrispondente all'indirizzo $034F.

Il programma, allora, continuerà ad essere eseguito dalla "parola" successiva (sempre all'indietro) a $034F, e cioè $034E ($A9). Bisogna pertanto prestare la massima attenzione al secondo byte delle istruzioni di BRANCH perchè, in caso di errore, potremmo far continuare il programma da un dato anzichè da un'istruzione o da un'istruzione indesiderata.

Notiamo facilmente che il blocco di istruzioni $034E-$035F srà eseguito 255 volte fino a che $0356 non conterrà $00, consentendo alla CPU di ignorare $D0 EE. Ogni volta che il blocco viene eseguito, lo schermo viene riempito di 256 simboli di "A". Poichè lo schermo è di 1000 caratteri, il blocco deve essere eseguito quasi quattro volte. Vediamo come.

PC=$0360: EE 52 03. Provvede ad incrementare il MSB del puntatore alla memoria schermo dopo che sono state eseguite le 255 iterazioni previste dal loop.

PC=$0363: EE 57 03. Vedi istruzione $0360 (riguarda la memoria colore).

PC=$0366: AD 52 03. Carica in accumulatore il dato contenuto in $0352.

PC=$0369: C9 08. Questa istruzione a due byte compara, mediante sottrazione, l'accumulatore con il numero $08; se i due numeri risultano uguali, fornirà $00 come risultato (attenzione: l'accumulatore non viene comunque alterato, nè tantomeno il numero $08).

PC=$036B: D0 E1. In questo caso, esattamente come prima ($035E), il primo bit del secondo byte è uno, e pertanto, nel caso in cui il risultato della comparazione $0369 sarà uguale a $00, il programma proseguirà all'istruzione successiva. Diversamente ci sarà un salto all'indietro fino alla locazione $034E.

PC=$036D: 60. Return from Subroutine: si esce dal L.M. e si ritorna al Basic.

Riepilogando, il blocco $034E-$035F viene eseguito 255 volte, mentre $034E-$036E viene eseguito 4 volte prima di tornare al Basic.

Il lettore potrà verificare la validità di questa routine in Linguaggio Macchina, in precedenza caricata, con il programmino suggerito qui sotto.

```
100 SYS 826
110 GET A$: IF A$="" THEN 110
120 A=ASC(A$): POKE 847,A
130 GOTO 100
```

Tiriamo le somme

Per assumere una certa familiarità con questo nuovo linguaggio, vediamo ora un altro esempio. Come abbiamo visto in precedenza, con le conoscenze che abbiamo, possiamo scrivere un numero intero compreso tra -32768 e +32767. Cerchiamo ora di capire come funziona il meccanismo della somma tra due numeri decimali e poi esadecimali. Facendo la somma dell'esempio qui sotto riportato, notiamo che quando la somma di due cifre corrispondenti supera il numero nove, dobbiamo considerare il riporto nella successiva colonna (somma tra 8 e 5; tra 6 e 4).

```
Decimale

1 1 (riporto)

16280 +
 4250 =
-------
20530
```

```
Esadecimale

1 (carry)

1A03 +
9B77 =
------
B57A
```

Analogamente avviene per due numeri esadecimali, solo che consideriamo il riporto quando la somma supera il numero $F (#15). Esempio: somma tra $A e $B nel caso sopra riportato. Prima di continuare, ricordiamo che, in inglese, riporto si traduce col termine CARRY, ed in seguito lo chiameremo sempre in questa maniera. Tra le istruzioni in L.M. del 6502 ve ne sono ben otto di somma tra due byte, e ne esaminiamo ora una.

Codice esadecimale $69: istruzione a due byte; codice mnemonico ADC (da ADd accumulator immediate whith Carry).
Esempio:

69 72
ADC #$72

Quando il microprocessore incontra questa istruzione, esegue la somma tra il valore che si trova in quel momento nell'accumulatore, il numero esadecimale $72 e l'eventuale Carry.

Vediamo ora alcuni esempi:

```
 10 REM PROGRAMMA 4
 15 :
 20 PRINTCHR$(147):AD=0
 30 PRINT"ADDIZIONA DUE NUMERI
    MINORI DI 255"
 40 PRINT"CONTENUTI IN #828 ED
    IN #830"
 50 PRINT"PONENDO IL RISULTATO
    IN #837"
100 READ A$:IF VAL(A$)<0 THEN P
    RINT"(VIENE ATTIVATO CON: S
    YS826)":PRINT:GOTO 170
110 X1=ASC(LEFT$(A$,1)):X2=ASC(
    RIGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
     140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
     160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+
    1:GOTO 100
170 INPUT "PRIMO ADDENDO IN 828
    ";PA:POKE 828,PA
180 INPUT "SECON.ADDENDO IN 830
    ";SA:POKE 830,SA:SYS826
190 PRINT"RISULTATO IN 837 (SYS
    826)="PEEK(837):PRINT:PRINT
    :GOTO 170
200 DATA 18,A9,02,69,07,8D,45,0
    3
210 DATA 60,EA,EA,00,-1
```

```
SOMMA TRA DUE BYTE
------------------

IND.N   ESA     DATI       MNEMON.
----------------------------------

 826   033A   18         CLC
 827   033B   A9 02      LDAIM 2
 829   033D   69 07      ADCIM 7
 831   033F   8D 45 03   STA 837
 834   0342   60         RTS
 835   0343   EA         NOP
 836   0344   EA         NOP
 837   0345   00         BRK
```

Analizziamo singolarmente istruzione per istruzione:

PC=$033A: 18. Clear Carry (CLC). Istruzione ad un solo byte, implicita, cioè non ha bisogno di indirizzi o di dati per la sua interpretazione. Questa istruzione cancella il Carry eventualmente presente nell'apposito registro della CPU. Ritorneremo in seguito su questa istruzione.

PC=$033B: A9 02. LoaD Accumulator con $02, come abbiamo già visto in precedenza.

PC=$033D: 69 07. Abbiamo già commentato un'istruzione di questo tipo.

PC=$033F: 8D 45 03. STore Accumulator in $0345, che viene usato come registro temporaneo.

PC=$0342: 60. ReTurn from Subroutine, già analizzata.
PC=$0343 : EA. NOP=No OPeration; trascriviamo questo gruppo di EA perchè in seguito sia più semplice rintracciare sullo schermo il risultato della somma nel caso si stia utilizzando un Monitor.

Per la cronaca, EA è un'istruzione implicita che significa: "Non eseguire alcuna operazione". Al contrario di quanto possa sembrare è un'istruzione molto utile, come vedremo in seguito.

Per far girare il programma, battiamo di seguito:

POKE 828,2: POKE830,7: SYS 826: PRINT PEEK (837)

Modificando opportunamente i valori delle due POKE, il risultato della PEEK conterrà sempre la somma dei due numeri in questione. In $0345 comparirà sempre il risultato della somma, a meno che la somma dei due numeri caricati non superi il valore $FF; infatti, consideriamo le seguenti somme esadecimali:

```
0C + 04 = 10  (nota bene)
40 + 0A = 4A
0B + 03 = 0E
FC + 14 = 10  (nota bene)
AB + 63 = 0E  (non corretto)
```

Infatti 0C = 12 dec.

04 = 04 dec.

04 + 12 = 16 dec. = 10 esa.

Come si può notare, allo stesso risultato si può arrivare anche sommando $FC e $14. Come facciamo ora a sapere se il valore contenuto in $0345 è il risultato di una somma superiore o inferiore a $FF ? A tale scopo consideriamo il registro di stato (ST) della CPU 6502: esso è un particolare registro ad 8 bit, interno alla CPU stessa, che memorizza diverse istruzioni tra le quali il Carry.

Il bit dell'ST corrispondente al Carry viene automaticamente settato (posto, cioè a 1) se, dopo l'esecuzione di alcune istruzioni, tra le quali la somma, si supera il valore di $FF. Da quel momento esso non viene più cancellato (cioè resettato o portato a 0) anche se, in seguito, si esegue una somma che non ha riporto.

Ecco perchè la prima istruzione di una procedura di somma deve SEMPRE essere CLC, che viene così interpretata: "Resetta il Carry, eventualmente posto ad 1 da una operazione precedente".

Il Carry

Miglioriamo ora il programma in L.M. visto e, per fare questo esaminiamo un'altra istruzione di Branch (salto condizionato):

BCS: Branch on Carry Set to 1 = Salta se il Carry è uguale ad 1. Codice Operativo: B0 xx, dove xx rappresenta l'entità del salto, come già verificato in precedenza.

```
10 REM PROGRAMMA 5
15 :
20 PRINTCHR$(147)
30 PRINT"ADDIZIONA DUE NUMERI M
   INORI DI 255"
40 PRINT"CONTENUTI IN #833 ED I
   N #835"
50 PRINT"PONENDO IL RISULTATO I
   N #851"
60 PRINT"E TENENDO CONTO DEL CA
   RRY IN #850"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"(SI ATTIVA CON: SYS826)"
    :PRINT:PRINT:GOTO 170
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+1
    :GOTO 100
170 INPUT "PRIMO ADDENDO";PA:POK
    E 833,PA
180 INPUT "SECO. ADDENDO";SA:POK
    E 835,SA:SYS826
190 PRINT"RISULTATO (IN 851)="PE
    EK(851)
195 PRINT"CARRY (IN 850)="PEEK(8
    50):PRINT:PRINT:GOTO 170
200 DATA 18,A9,00,8D,52,03,A9,02
210 DATA 69,07,8D,53,03,B0,01,60
220 DATA A9,01,8D,52,03,60,EA,EA
230 DATA 00,00,-1
```

SOMMA CON RIPORTO

IND.N	ESA	DATI	MNEMON.
826	033A	18	CLC
827	033B	A9 00	LDAIM 0
829	033D	8D 52 03	STA 850
832	0340	A9 02	LDAIM 2
834	0342	69 07	ADCIM 7
836	0344	8D 53 03	STA 851
839	0347	B0 01	BCS 1
841	0349	60	RTS
842	034A	A9 01	LDAIM 1
844	034C	8D 52 03	STA 850
847	034F	60	RTS
848	0350	EA	NOP
849	0351	EA	NOP
850	0352	00	BRK
851	0353	00	BRK

Vediamo ora di analizzare le istruzioni del programma.

A9 00 e 8D 52 03. Queste due istruzioni scrivono $00 nella locazione $0352 che servirà a visualizzare l'eventuale riporto.

B0 01. Se la somma appena eseguita supera il valore $FF, il bit di carry viene posto a 1, e pertanto il salto di un byte è eseguito e l'elaborazione continua da $034A.

60. Se la somma eseguita non ha riporto, il Carry rimane come prima, e cioè a 0 (cfr. istruzione $033A 18); si ritorna al Basic.

A9 01 e 8D 52 03. Nella locazione $0352 viene trascritto il valore $01, ad indicare l'avvenuto riporto.

Facciamo ora girare il programma per controllarne la funzionalità:

POKE 833,primo addendo: POKE 835,secondo addendo: SYS 826: PRINT PEEK(850),PEEK(851)

Il primo valore (zero o uno) indicherà la presenza di un eventuale riporto, mentre il secondo costituirà la somma tra i due addendi inseriti tramite le due POKE.

Eseguiamo ora una somma di due numeri, ciascuno dei quali occupa due byte.

Decimale	Esadecimale
11290 +	2CEA +
14891 =	3221 =
-------	------
26181	5F1B

In pratica, dobbiamo dapprima eseguire la somma tra gli ultimi due byte (EA+21), trascrivere il risultato da parte e tenere presente l'eventuale Carry (nel caso specifico esiste). In seguito, sommiamo i primi due byte tra di loro considerando anche l'eventuale Carry dell'addizione precedente (2C+32+1). Il programma può essere il seguente:

```
10 REM PROGRAMMA 6
15 :
20 PRINTCHR$(147)
30 PRINT"SOMMA DUE NUMERI MINOR
   I DI 65535"
40 PRINT"CONTENUTI NELLA FORMA"
50 PRINT"LOW BYTE - HIGH BYTE"
60 PRINT"1 ADDENDO : #828 - #83
   5"
70 PRINT"2 ADDENDO : #830 - #83
   7"
80 PRINT"RISULTATO : #842 - #84
   3"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"(SI ATTIVA CON: SYS826)"
    :PRINT:PRINT:GOTO 170
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+1
    :GOTO 100
170 INPUT "L/H BYTE PRIMO ADDEND
    O";P1,P2:POKE 828,P1:POKE 83
    5,P2
180 INPUT "L/H BYTE SECON ADDEND
    O";S1,S2:POKE 830,S1:POKE 83
    7,S2:SYS826
190 PRINT"RISULTATO (L/H):"PEEK(
    842)PEEK(843):PRINT:PRINT:GO
    TO 170
200 DATA 18,A9,EA,69,21,8D,4A,03
210 DATA A9,2C,69,32,8D,4B,03,60
220 DATA 00,00,-1
```

SOMMA CON QUATTRO BYTE

IND.N	ESA	DATI	MNEMON.
826	033A	18	CLC
827	033B	A9 EA	LDAIM 234
829	033D	69 21	ADCIM 33
831	033F	8D 4A 03	STA 842
834	0342	A9 2C	LDAIM 44
836	0344	69 32	ADCIM 50
838	0346	8D 4B 03	STA 843
841	0349	60	RTS
842	034A	00	BRK
843	034B	00	BRK

Prima di eseguire la seconda somma (LDA #$2C -ADC #$32) non bisogna inserire CLC, perchè è necessario considerare il Carry, eventual-

mente presente in seguito alla prima somma. Il lettore, per esercizio, può modificare il programma, inserendo una "spia" per l'eventuale secondo riporto, come abbiamo fatto nel programmino precedente.

Parliamo ora di un'altra istruzione implicita: SED (SEt Decimal mode), codice operativo F8. Dall'istruzione successiva a SED le somme vengono eseguite in decimale anzichè in esadecimale: $F8 è di notevole comodità in alcuni casi; provate ad inserirla nei programmi precedenti o subito dopo. Attenzione però che se il modo di operare è decimale e chiediamo di sommare valori esadecimali si verificano errori. Per ripristinare il modo esadecimale è necessaria l'istruzione implicita CLD (CLear Decimal mode), codice operativo $D8; senza questa istruzione, infatti, la CPU continuerà a ragionare in decimale. Consiglio a chi desidera sperimentare l'istruzione SED di inserire D8 prima di ogni RTS per evitare malfunzionamenti del Basic.

Problemi di gestione di un programma

Finora abbiamo cercato di fare un po' di luce sulla gestione interna del 6502, microprocessore utilizzato, come ben sappiamo, dal Commodore 64 e dal C128 nella versione 64. Ma quello che conosciamo non permette ancora di creare programmi di una certa utilità ed interesse o, per lo meno, richiede sforzi particolari ed una notevole quantità di memoria. Si tratta di approfondire il discorso relativo agli indirizzamenti. Senza alcuna pretesa di riproporre ciò che è già stato detto, vale la pena citare i quattro indirizzamenti che conosciamo affiancati da altrettanti esempi:

- Implicito (AA = TAX)
- Immediato (A9 00 = LDA #$00)
- Assoluto (8D 00 04 = STA $0400)
- Relativo (D0 03 = BNE +$03)

Questi consentono di leggere il contenuto di una locazione di memoria, di modificarla e di eseguire salti condizionati. Proviamo a creare ora un semplice programmino che trasferisca sullo schermo un messaggio i cui valori ASCII sono contenuti in memoria. Il programma in questione è numerato con 7.

```
10 REM PROGRAMMA 7
15 :
20 PRINTCHR$(147)CHR$(14)
30 PRINT"FA APPARIRE IN ALTO A
   SINISTRA"
40 PRINT"SULLO SCHERMO UNA SCRI
   TTA"
50 PRINT"CONTENUTA IN MEMORIA"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS826":EN
    D
110 X1-ASC(LEFT$(A$,1)):X2-ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1-X1-55:GOTO
    140
130 X1-X1-48
140 IF X2>57 THEN X2-X2-55:GOTO
    160
150 X2-X2-48
160 POKE 826+AD,X1*16+X2:AD-AD+1
    :GOTO 100
200 DATA A9,04,8D,5C,03,A9,03,8D
210 DATA 59,03,A9,71,8D,58,03,A9
220 DATA D8,8D,61,03,A9,00,8D,5B
230 DATA 03,8D,60,03,AA,AD,71,03
240 DATA 8D,00,04,A9,00,8D,00,D8
250 DATA EE,58,03,EE,5B,03,EE,60
260 DATA 03,E8,E0,18,D0,E7,60
265 REM MESSAGGIO DA FAR APPARIR
    E
270 DATA 43,4F,4D,4D,4F,44,4F,52
    ,45
280' DATA 20,43,4F,4D,50,55,54,45
290 DATA 52,20,43,4C,55,42,20,00
    ,-1
```

```
VISUALIZZA MESSAGGIO
--------------------

IND.N   ESA    DATI        MNEMON.
----------------------------------

 826   033A   A9 04       LDAIM 4
 828   033C   8D 5C 03    STA 860
 831   033F   A9 03       LDAIM 3
 833   0341   8D 59 03    STA 857
 836   0344   A9 71       LDAIM 113
 838   0346   8D 58 03    STA 856
 841   0349   A9 D8       LDAIM 216
 843   034B   8D 61 03    STA 865
 846   034E   A9 00       LDAIM 0
 848   0350   8D 5B 03    STA 859
 851   0353   8D 60 03    STA 864
 854   0356   AA          TAX
 855   0357   AD 89 03    LDA 905
 858   035A   8D 18 04    STA 1048
 861   035D   A9 00       LDAIM 0
 863   035F   8D 18 D8    STA 55320
 866   0362   EE 58 03    INC 856
 869   0365   EE 5B 03    INC 859
 872   0368   EE 60 03    INC 864
 875   036B   E8          INX
 876   036C   E0 18       CPXIM 24
 878   036E   D0 E7       BNE 231
 880   0370   60          RTS
```

La parte iniziale (da $033A a $0356) costituisce la necessaria inizializzazione del programma. Il perchè verrà spiegato in seguito. Successivamente, l'accumulatore viene caricato con il contenuto della locazione di memoria a partire dalla quale sono inseriti i dati. Quindi il dato (in codice ASCII) viene posto nella casella in alto a sinistra nello schermo. Le successive due istruzioni (A9 00 e 8D 00 D8) settano il colore del carattere appena visualizzato.

Per rendere ciclico il programma, si deve adesso modificare il Low Byte dei tre registri che fanno diretto riferimento alla memoria ed allo schermo (EE 58 03, ecc.). Il loop si chiude con un paragone, necessario per valutare il numero esatto di caratteri da trasferire, e con un salto in caso di mancato raggiungimento di tale numero (E0 18 e D0 E7). Si capisce quindi il perchè si renda necessaria l'inizializzazione, senza la quale il programma, che in pratica si automodifica, non avrebbe potuto girare più di una volta consecutivamente senza ingenerare errori e cattivo funzionamento.

L'indirizzamento indicizzato

Il programma non è certo semplice, ma gli "architetti" del 6502 hanno previsto tale evenienza e dotato il microprocessore di un particolare indirizzamento: l'indirizzamento indicizzato. Il programma che proponiamo (N.8), esegue lo stesso lavoro del programma appena descritto, ma in modo più semplice e decisamente più breve.

Il "cuore" è costituito dalle due nuove istruzioni BD nn nn e 9D nn nn (LDA $nnnn,X e STA $nnnn,X); la loro particolarità è quella di sommare al valore assoluto che segue l'istruzione stessa il valore del registro X (registro ad 8 bit analogo ad A).

Per esempio, supponiamo di avere il seguente programma:

```
LM              ASSEMBLER

A2 05           LDX #$05
BD 00 04        LDA $0400,X
60              RTS
```

ne di tutti questi, cosicchè alla resa dei conti abbiamo un set di 151 istruzioni con tutte le varianti possibili.

A seconda dell'operazione che svolgono, le istruzioni si dividono in:

• TRASFERIMENTO DATI: caricano i registri con la memoria (LDA, LDX, LDY); caricano la memoria con i registri (STA, STX, STY); interscambiano i registri tra di loro (TAX, TAY, TYA, TXA)

• ELABORANO I DATI: eseguono le funzioni aritmetiche e logiche (ADC, SBC, AND, ORA, EOR); incrementano e decrementano registri e memoria (INC, DEC, INX, DEX, INY, DEY)
• CONFRONTI, TEST & DIRAMAZIONI: eseguono confronti fra registri e memoria (CPX, CPY, CMP); eseguono diramazioni in seguito a confronti con il registro di stato P (BMI, BPL, BCC, BCS, BEQ, BNE, BVS, BVC)
• CONTROLLO: modificano il registro di stato P (CLC, SEC, CLD, SED, CLV)
• FLUSSO DI PROGRAMMA: eseguono salti con o senza ritorno (JMP, JSR, RTS, RTI)

Analizzate le istruzioni di base, ricapitoliamo i medodi di indirizzamento, quelli cioè che determinano il campo nel quale agisce una determinata istruzione:

• IMPLICITO (1 byte) riguarda per lo più la gestione interna dei registri (TAX, INX), ma si può occupare anche della gestione del flag di stato P (CLC) e dell'Interrupt (CLI, SEI); non richiede in nessun caso che vengano specificati parametri.
• IMMEDIATO (2 byte) l'operazione indicata dal codice operativo viene eseguita sul numero che lo segue immediatamente (come: LDA #$00); l'operando, naturalmente, non può essere maggiore di 255.
• ASSOLUTO (3 byte) l'operando che deve essere "trattato" dall'istruzione è contenuto nel registro di memoria identificato dai 2 byte che seguono il codice operativo stesso (LDA $A100); i due byte sono nella solita disposizione LB, HB.
• PAGINA ZERO (2 byte) variante dell'indirizzamento assoluto, pone automaticamente HB uguale a 0, limitando così il campo dell'operazione alle prime 256 locazioni di memoria che, peraltro, contengono dati molto importanti per il funzionamento del calcolatore (LDA $F0); tale restrizione aumenta la velocità di esecuzione.
• RELATIVO (2 byte) la parola che segue l'istruzione indica uno spostamento, in avanti di 128 parole ed all'indietro di 127, secondo uno standard piuttosto complesso (BEQ $+12); in ogni caso provvede ai debiti calcoli l'assemblatore.
• INDICIZZATO (2 byte in pagina zero oppure 3 byte se assoluto) l'indirizzo sul quale deve agire l'istruzione viene ottenuto dalla somma del dato (o dei dati se assoluto) posto dopo il codice operativo, con il registro interno (solo X e Y) specificato nell'istruzione stessa (LDA $A001,Y); se abbiamo per esempio l'istruzione LDA $C000,Y ,dove Y=$05, la locazione di memoria che verrà posta in A sarà $(C000+05)=$C005.

L'indirizzamento indiretto

Vediamo ora di fare conoscenza con l'indirizzamento indiretto.
Il termine "indiretto" tecnicamente deriva dal fatto che invece di prelevare l'indirizzo che segue immediatamente l'istruzione, il successivo valore nella sequenza di programma è un puntatore all'indirizzo interessato. Chiariamo meglio il concetto.

Tutti sanno che nelle pagine 0,1,2,3 (cioè nelle locazioni di memoria che vanno da 0 a 1023, appena sotto il video) sono contenuti un gran numero di puntatori, che "puntano", per l'appunto (!), a routine nelle ROM fondamentali per il funzionamento del Commodore 64. E' anche noto che questo fatto aumenta notevolmente le doti di flessibilità del calcolatore, perchè permette all'utente di modificare a proprio piacimento tutti o quasi i procedimenti normalmente eseguiti automaticamente. A questo punto è lecito chiedersi come faccia il 6502 a prelevare l'indirizzo indicato nei puntatori. In effetti, con le conoscenze che abbiamo a disposizione, questo non è permesso, a meno di complicazioni assurde che rasentano il limite della correttezza e della linearità di programmazione.

Ma gli "architetti" del 6502 hanno pensato anche a questo, creando un'unica istruzione che sfrutti l'indirizzamento indiretto non indicizzato (più oltre analizzeremo l'indicizzazione abbinata all'indirezione), ma che si rivela di importanza fondamentale soprattutto nella gestione della ROM cui accennavamo prima: si tratta dell'istruzione di salto indiretto JMP (XXXX) (codice operativo: 6C XX XX). Il suo funzionamento è piuttosto semplice da capire, ma occorre prestare sempre una certa attenzione ai calcoli da eseguire in fase di stesura del programma e di modifica dei puntatori. Infatti in seguito a questa istruzione il P.C. (Contatore di Programma) viene modificato in modo tale che il Low Byte sia caricato con la parola contenuta nella locazione di memoria indicata dall'istruzione, e che l' High Byte sia caricato con il contenuto della locazione immediatamente seguente a quella indicata.

In altre parole data l'istruzione JMP($0306) e supponendo che $0306 contenga il dato $1A e che $0307 contenga il dato $A7, il P.C. si posizionerà all'indirizzo $A714 (vettore di List).

L'utilità pratica del fatto di avere in un'area RAM (e quindi modificabile) tutti i vettori relativi alle funzioni più importanti è basilare, perchè permette di alterare tali routines, implementandole o addirittura sostituendole con altre create dall'utente.

Accennavamo prima alla possibilità di abbinare l'indicizzazione alla tecnica dell'indirezione. Fin d'ora è bene chiarire che esistono due soluzioni ben differenti fra loro, nonostante la buffa (ma non troppo) caratteristica di chiamarsi pressappoco allo stesso modo (indirizzamento indicizzato indiretto e indiretto indicizzato). La caratteristica pratica che li contraddistingue è che la prima di queste tecniche sfrutta solo ed esclusivamente l'indice Y, mentre la seconda quello X. Prima di osservare le ulteriori differenze, è indispensabile

```
10 REM PROGRAMMA 8
15 :
20 PRINTCHR$(147);CHR$(14)
30 PRINT"FA APPARIRE IN ALTO A
   SINISTRA"
40 PRINT"SULLO SCHERMO UNA SCRI
   TTA"
50 PRINT"CONTENUTA IN MEMORIA"
60 PRINT"SFRUTTANDO L'INDICIZZA
   ZIONE"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS826":EN
    D
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+1
    :GOTO 100
200 DATA A2,00,BD,4D,03,9D,00,04
210 DATA A9,00,9D,00,D8,E8,E0,18
220 DATA D0,F0,60
221 REM MESSAGGIO DA VISUALIZZAR
    E
225 DATA 43,4F,4D,4D,4F
230 DATA 44,4F,52,45,20,43,4F,4D
240 DATA 50,55,54,45,52,20,43,4C
250 DATA 55,42,20,00,-1
```

MESSAGGIO INDICIZZATO

IND.N	ESA	DATI	MNEMON.
826	033A	A2 00	LDXIM 0
828	033C	BD 4D 03	LDAX 845
831	033F	9D 00 04	STAX 1024
834	0342	A9 00	LDAIM 0
836	0344	9D 00 D8	STAX 55296
839	0347	E8	INX
840	0348	E0 18	CPXIM 24
842	034A	D0 F0	BNE 240
844	034C	60	RTS

L'accumulatore verrà caricato con il contenuto della locazione \$(0400+05)=\$0405. Dunque, il registro X viene sommato al valore assoluto, e solo allora viene letta la locazione risultante. Analogo ragionamento vale per 9D nn nn, che, invece di leggere, modifica il contenuto della locazione risultante.

A tutto questo discorso aggiungiamo l'incremento e la comparazione del registro X con il valore assoluto ed otteniamo il programma in questione. Due aspetti importanti da sottolineare sono i seguenti:

• Il valore assoluto dell'indirizzamento rimane sempre lo stesso; quindi il programma non si automodifica e non ha bisogno di inizializzazione.
• Il 6502 tiene conto anche del riporto, vale a dire che, se nel corso del programma si supera una pagina di memoria, l'indicizzazione provvede a tutto. Esempio: \$(7BFE+05) dà automaticamente \$7C04.

Per quanto riguarda il trasferimento di blocchi con più di 256 locazioni di memoria, si può dare un'occhiata al programma N.9.

```
10 REM PROGRAMMA 9
15 :
20 PRINTCHR$(147);CHR$(14)
30 PRINT"STAMPA SULLO SCHERMO D
   EL"
40 PRINT"COMMODORE 64 LE ISTRUZ
   IONI"
50 PRINT"BASIC ED I MESSAGGI DI
    ERRORE"
60 PRINT"SFRUTTANDO L'INDICIZZA
   ZIONE"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS826":EN
    D
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 826+AD,X1*16+X2:AD=AD+1
    :GOTO 100
200 DATA A9,04,8D,78,03,A9,A0,8D
210 DATA 78,03,A9,D8,8D,80,03,A9
220 DATA 00,8D,77,03,8D,7A,03,8D
230 DATA 7F,03,A0,03,A2,00,20,76
240 DATA 03,E8,D0,FA,EE,78,03,EE
250 DATA 78,03,EE,80,03,88,D0,EC
260 DATA 20,76,03,A2,E8,20,76,03
270 DATA CA,D0,FA,60,BD,00,A0,9D
280 DATA 00,04,A9,00,9D,00,D8,60
    ,-1
```

Questo provvede a trasferire sullo schermo le istruzioni del Basic ed i relativi messaggi di errore. La sua struttura non è molto differente da quella dei programmi appena commentati, pur sfruttando l'indicizzazione. La differenza è che invece di alterare il Low Byte della locazione di memoria interessata, viene fatto variare lo High Byte.

Per riempire lo schermo del C/64, sono necessari 3 cicli da 256 byte l'uno, più un semiciclo (chiamato di offset) da 232 byte ($256*3+232=1000$). Per il conto dei cicli viene usato il registro Y (analogo ad A e X), che viene decrementato ogni 256 byte trasferiti, mentre l'inizializzazione si rende necessaria per impostare i corretti valori negli High Byte modificati dal programma stesso. L'istruzione 20 nn nn (JSR \$nnnn) consente di fare un salto ad un determinato sottoprogramma e di ritornare al punto di partenza quando incontra l'istruzione \$60 (RTS). Ma anche questi limiti possono facilmente essere superati ed in seguito ne vedremo il metodo.

Un breve riepilogo

In precedenza abbiamo analizzato le istruzioni più importanti ed alcuni indirizzamenti tipici del 6502. Prima di riesaminarle molto velocemente, è bene ricordare che le istruzioni sono in tutto 56, mentre gli indirizzamenti disponibili sono 13; comunque nessuna istruzione prevede l'utilizzazio-

VISUALIZZA ISTRUZIONI BASIC

IND.N	ESA	DATI	MNEMON.	IND.N	ESA	DATI	MNEMON.
826	033A	A9 04	LDAIM 4	862	035E	EE 78 03	INC 891
828	033C	8D 78 03	STA 891	865	0361	EE 78 03	INC 888
831	033F	A9 A0	LDAIM 160	868	0364	EE 80 03	INC 896
833	0341	8D 78 03	STA 888	871	0367	88	DEY
836	0344	A9 D8	LDAIM 216	872	0368	D0 EC	BNE 236
838	0346	8D 80 03	STA 896	874	036A	20 76 03	JSR 886
841	0349	A9 00	LDAIM 0	877	036D	A2 E8	LDXIM 232
843	034B	8D 77 03	STA 887	879	036F	20 76 03	JSR 886
846	034E	8D 7A 03	STA 890	882	0372	CA	DEX
849	0351	8D 7F 03	STA 895	883	0373	D0 FA	BNE 250
852	0354	A0 03	LDYIM 3	885	0375	60	RTS
854	0356	A2 00	LDXIM 0	886	0376	BD 00 A0	LDAX 40960
856	0358	20 76 03	JSR 886	889	0379	9D 00 04	STAX 1024
859	035B	E8	INX	892	037C	A9 00	LDAIM 0
860	035C	D0 FA	BNE 250	894	037E	9D 00 D8	STAX 55296
				897	0381	60	RTS

osservare che entrambe "lavorano" solo in pagina zero, aspetto questo da tenere bene a mente.

L'indicizzato indiretto

L'indirizzamento indicizzato indiretto (quello che sfrutta il registro Y, per intenderci) si comporta nel modo seguente:
l'indirizzo effettivo dell'istruzione è dato dal contenuto della locazione in pagina 0 (L.B.), più il contenuto della locazione successiva moltiplicato per 256 (H.B.), più il registro Y (indice). In altre parole, i due bytes in pagina 0 (dei quali viene indicato solo il primo), sono dei puntatori ad una tabella di 256 elementi locata in qualsiasi area della memoria, che può essere scorsa per intero incrementando o decrementando (INY, DEY) solo l'indice Y.

Esempio: LDA($F0),Y
Se la locazione $F0 contiene il dato $16 e $F1 contiene $C0, e Y vale $03, carica l'accumulatore con il contenuto della locazione di memoria $ (C016+03)=$C019.

L'indiretto indicizzato

Nel caso dell'indirizzamento indiretto indicizzato (con X come indice), le cose sono lievemente diverse. L'indice, infatti, viene aggiunto immediatamente all'indirizzo specificato dall'istruzione, per puntare ad un altro indirizzo indiretto (espresso nella forma: LB, HB) sempre in pagina 0. E' importante rilevare che la somma non tiene conto del riporto, quindi se il risultato supera il limite della pagina la locazione considerata è quella risultante diminuita di 256. Tale caratteristica permette di creare una tabella di puntatori allocati in pagina 0 che può essere analizzata modificando semplicemente l'indice X (INX, DEX).

Esempio: LDA($F0,X) carica in A la "parola" (dato) il cui indirizzo è puntato dal contenuto della coppia di locazioni $(F0+X), come low byte, e $(F0+1+X), come high byte.

Dunque, ricapitolando, l'indice Y influisce sull'indirizzo effettivo, mentre l'indice X modifica direttamente i puntatori in pagina 0; ed ecco spiegata la strana somiglianza dei due nomi. Dall'uso contemporaneo di entrambe le tecniche risultano vantaggi immediatamente apprezzabili che permettono finalmente di lavorare in linguaggio macchina con linearità e disinvoltura.

Applicazioni

Per quanto riguarda le applicazioni, credo che i programmi proposti si commentino da soli, ma li analizzeremo, comunque, separatamente.

```
 10 REM  PROGRAMMA 10
 20 PRINTCHR$(147),CHR$(14)
 30 PRINT"TRASFERISCE SULLO SCHE
    RMO DEL"
 40 PRINT"COMMODORE 64 LE ISTRUZ
    IONI BASIC ED"
 50 PRINT"I MESSAGGI DI CONTROLL
    O "
 60 PRINT"SFRUTTANDO L'INDIRIZZA
    MENTO INDICIZZATO"
 70 PRINT"INDIRETTO (CON Y)"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS49152":
    END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=AD
    +1:GOTO 100

200 DATA A9,04,85,62,A9,A0,85,64
210 DATA A9,D8,85,66,A9,00,85,61
220 DATA 85,63,85,65,A2,03,A0,00
230 DATA 20,23,C0,A2,01,A0,E8,20
240 DATA 23,C0,60,A9,00,91,65,B1
250 DATA 63,91,61,88,D0,F5,E6,62
260 DATA E6,64,E6,66,CA,D0,EC,60
    ,-1
```

Il programma 10 esegue esattamente la stessa operazione dell'ultimo programma presentato in precedenza, e cioè trasferisce i messaggi di errore ed i tokens del Basic sul video, solo che sfrutta la tecnica dell'indirizzamento indicizzato indiretto (Y). La prima parte costituisce la necessaria inizializzazione, che permette di ripetere più volte l'esecuzione senza incorrere in grossolani errori. Tutti i valori interessati ($A000, $0400, $D800) vengono sistemati in pagina 0; nel nostro caso vengono locati in un'area destinata al floating point (FLP), e che quindi va adoperata solo quando non si usa il FLP. In ogni caso a lato è riportata una tabella di locazioni libere o per lo meno utilizzabili in pagina 0.

Il registro X indica il numero di blocchi (da 256 bytes l'uno) che devono essere trasferiti, mentre il registro Y contiene l'eventuale OFFSET, cioè il numero di bytes (inferiori a 256) che devono essere trattati oltre ai precedenti blocchi. La subroutine si è resa necessaria per meglio identificare la parte centrale del programma, che del resto è già chiara per conto suo: analizzandola ci si rende effettivamente conto della funzione di X come contatore per 256 e della funzione di Y come indice.

VISUALIZZA MESSAGGI (INDIR. INDIC. INDIRETTO)

IND.N	ESA	DATI	MNEMON.	IND.N	ESA	DATI	MNEMON.
49152	C000	A9 04	LDAIM 4	49183	C01F	20 23 C0	JSR 49187
49154	C002	85 62	STAZ 98	49186	C022	60	RTS
49156	C004	A9 A0	LDAIM 160	49187	C023	A9 00 .	LDAIM 0
49158	C006	85 64	STAZ 100	49189	C025	91 65	STAIY 101
49160	C008	A9 D8	LDAIM 216	49191	C027	B1 63	LDAIY 99
49162	C00A	85 66	STAZ 102				
49164	C00C	A9 00	LDAIM 0	49193	C029	91 61	STAIY 97
49166	C00E	85 61	STAZ 97	49195	C02B	88	DEY
49168	C010	85 63	STAZ 99	49196	C02C	D0 F5	BNE 245
49170	C012	85 65	STAZ 101	49198	C02E	E6 62	INCZ 98
49172	C014	A2 03	LDXIM 3	49200	C030	E6 64	INCZ 100
49174	C016	A0 00	LDYIM 0	49202	C032	E6 66	INCZ 102
49176	C018	20 23 C0	JSR 49187	49204	C034	CA	DEX
49179	C01B	A2 01	LDXIM 1	49205	C035	D0 EC	BNE 236
49181	C01D	A0 E8	LDYIM 232	49207	C037	60	RTS

```
10 REM PROGRAMMA 11
15 :
20 PRINTCHR$(147),CHR$(14)
30 PRINT"TRASFERISCE SULLO SCHE
   RMO DEL "
40 PRINT"COMMODORE 64 LE ISTRUZ
   IONI BASIC ED"
50 PRINT"I MESSAGGI DI CONTROLL
   O "
60 PRINT"SFRUTTANDO LE ROUTINES
    DELLA _|\"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS49152":
    END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=AD
    +1:GOTO 100
200 DATA A9,00,85,5F,A9,A0,85,60
210 DATA A9,E8,85,5A,A9,A3,85,5B
220 DATA A9,E8,85,58,A9,07,85,59
230 DATA 20,BF,A3,60,-1
```

VISUALIZZA MESSAGGI
CON ROUTINE ROM

IND.N	ESA	DATI	MNEMON.
49152	C000	A9 00	LDAIM 0
49154	C002	85 5F	STAZ 95
49156	C004	A9 A0	LDAIM 160
49158	C006	85 60	STAZ 96
49160	C008	A9 E8	LDAIM 232
49162	C00A	85 5A	STAZ 90
49164	C00C	A9 A3	LDAIM 163
49166	C00E	85 5B	STAZ 91
49168	C010	A9 E8	LDAIM 232
49170	C012	85 58	STAZ 88
49172	C014	A9 07	LDAIM 7
49174	C016	85 59	STAZ 89
49176	C018	20 BF A3	JSR 41919
49179	C01B	60	RTS

• Il programma 11 non è altro che una variazione sul tema del programma precedente. La sua caratteristica è quella di sfruttare una subroutine già presente nelle ROM del C/64 che effettua il trasferimento dei dati ($A3BF). Tale procedura richiede, come dati in ingresso: la locazione d'inizio dei dati da trasferire (5F, 60), la locazione finale dei dati da trasferire (5A, 5B) e la locazione finale della destinazione dei dati (58, 59). I valori in questione vengono forniti dalla parte iniziale, quindi si salta direttamente in ROM per tornare al Basic alla fine dell'esecuzione.

```
10 REM PROGRAMMA 12
15 :
20 PRINTCHR$(147),CHR$(14)
30 PRINT"IMPOSTA E PULISCE LA P
   AGINA GRAFICA"
40 PRINT"SFRUTTANDO L'INDIRIZZA
   MENTO INDICIZZATO"
50 PRINT"INDIRETTO (IN Y), QUIN
   DI DISEGNA UNA"
60 PRINT"SINUSOIDE"
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS49152:R
    UN170":END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=AD
    +1:GOTO 100
170 FOR X=0 TO 319
180 V=INT(100+90*SIN(X/51))
190 CH=INT(X/8):RO=INT(V/8)
200 LN=V AND 7:BI=7-(X AND 7)
210 BY=8192+RO*320+8*CH+LN
220 POKE BY,PEEK(BY) OR (2↑BI)
230 NEXT:POKE 1024,16
240 GOTO 240
300 DATA AD,18,D0,09,08,8D,18,D0
310 DATA AD,11,D0,09,20,8D,11,D0
320 DATA A9,40,85,FD,A9,20,85,FC
330 DATA A9,00,85,FB,A8,20,2E,C0
340 DATA A9,08,85,FD,A9,04,85,FC
350 DATA A9,03,20,2E,C0,60,91,FB
360 DATA 88,D0,FB,E6,FC,A6,FC,E4
370 DATA FD,D0,F3,60,-1
```

DISEGNO SINUSOIDE HI-RES

IND.N	ESA	DATI	MNEMON.
49152	C000	AD 18 D0	LDA 53272
49155	C003	09 08	ORAIM 8
49157	C005	8D 18 D0	STA 53272
49160	C008	AD 11 D0	LDA 53265
49163	C00B	09 20	ORAIM 32
49165	C00D	8D 11 D0	STA 53265
49168	C010	A9 40	LDAIM 64
49170	C012	85 FD	STAZ 253
49172	C014	A9 20	LDAIM 32
49174	C016	85 FC	STAZ 252
49176	C018	A9 00	LDAIM 0
49178	C01A	85 FB	STAZ 251
49180	C01C	A8	TAY
49181	C01D	20 2E C0	JSR 49198
49184	C020	A9 08	LDAIM 8
49186	C022	85 FD	STAZ 253
49188	C024	A9 04	LDAIM 4
49190	C026	85 FC	STAZ 252
49192	C028	A9 03	LDAIM 3
49194	C02A	20 2E C0	JSR 49198
49197	C02D	60	RTS
49198	C02E	91 FB	STAIY 251
49200	C030	88	DEY
49201	C031	D0 FB	BNE 248
49203	C033	E6 FC	INCZ 252
49205	C035	A6 FC	LDXZ 252
49207	C037	E4 FD	CPXZ 253
49209	C039	D0 F3	BNE 243
49211	C03B	60	RTS

• Il programma 12, inserito per scopi puramente dimostrativi, è una semplice impostazione della pagina grafica e della sua pulizia, ricalcando esattamente la procedura Basic: l'unica differenza? Provate a cronometrare quanto tempo ci mette, se ci riuscite!

• Il programma 13 è un'applicazione, indubbiamente futile e banale, di entrambe le tecniche di indicizzazione. La prima parte serve per inizializzare i puntatori, trasferendo in pagina zero i sei byte che seguono immediatamente il programma stesso. Per fare ciò si sfrutta l'indicizzazione indiretta, anche se obiettivamente se ne potrebbe fare a meno; ciononostante è importante rilevare la tecnica utilizzata, vale a dire piazzare i puntato-

```
10 REM PROGRAMMA 13
15 :
20 PRINTCHR$(147),CHR$(14)
30 PRINT"SCRIVE SULLO SCHERMO 3
    MESSAGGI"
40 PRINT"DIFFERENTI A SECONDA S
   I PREMA 1, 2 O 3"
50 PRINT"E TERMINA CON 0 ."
60 PRINT"SFRUTTA ENTRAMBE LE TE
   CNICHE DI"
70 PRINT"INDIREZIONE INDICIZZAT
   A. ATTENDERE..."
100 READ A$:IF VAL(A$)<0 THEN PR
    INT"ATTIVARE CON: SYS49152":
    END
110 X1=ASC(LEFT$(A$,1)):X2=ASC(R
    IGHT$(A$,1))
120 IF X1>57 THEN X1=X1-55:GOTO
    140
130 X1=X1-48
140 IF X2>57 THEN X2=X2-55:GOTO
    160
150 X2=X2-48
160 POKE 49152+AD,X1*16+X2:AD=AD
    +1:GOTO 100
200 DATA A9,19,85,FB,A9,00,85,FC
210 DATA A9,41,85,FD,A9,C0,85,FE
220 DATA A0,05,B1,FD,91,FB,88,10
230 DATA F9,20,87,EA,20,3E,F1,38
240 DATA E9,30,F0,1C,30,F3,C9,04
250 DATA B0,EF,0A,AA,A0,0D,A1,17
260 DATA 20,CA,F1,F6,17,88,10,F6
270 DATA A9,8D,20,CA,F1,4C,00,C0
    ,60
272 REM PUNTATORI PRIMO MESSAGGI
    O
274 DATA 7D,E4
276 REM PUNTATORI SECONDO MESSAG
    GIO
278 DATA 98,E4
279 REM PUNTATORI TERZO MESSAGGI
    O
280 DATA 47,C0
285 REM TERZO MESSAGGIO
290 DATA 42,45,54,54,4F,4C,41,20
    ,53
300 DATA 49,4D,4F,4E,45,-1
```

```
VISUALIZZA TRE MESSAGGI
-----------------------

IND.N   ESA     DATI        MNEMON.
-----------------------------------

49152   C000   A9 19        LDAIM 25
49154   C002   85 FB        STAZ 251
49156   C004   A9 00        LDAIM 0
49158   C006   85 FC        STAZ 252
49160   C008   A9 41        LDAIM 65
49162   C00A   85 FD        STAZ 253
49164   C00C   A9 C0        LDAIM 192
49166   C00E   85 FE        STAZ 254
49168   C010   A0 05        LDYIM 5
49170   C012   B1 FD        LDAIY 253
49172   C014   91 FB        STAIY 251
49174   C016   88           DEY
49175   C017   10 F9        BPL 249
49177   C019   20 87 EA     JSR 60039
49180   C01C   20 3E F1     JSR 61758
49183   C01F   38           SEC
49184   C020   E9 30        SBCIM 48
49186   C022   F0 1C        BEQ 28
49188   C024   30 F3        BMI 243
49190   C026   C9 04        CMPIM 4
49192   C028   B0 EF        BCS 239
49194   C02A   0A           ASLA
49195   C02B   AA           TAX
49196   C02C   A0 0D        LDYIM 13
49198   C02E   A1 17        LDAIX 23
49200   C030   20 CA F1     JSR 61898
49203   C033   F6 17        INCZX 23
49205   C035   88           DEY
49206   C036   10 F6        BPL 246
49208   C038   A9 8D        LDAIM 141
49210   C03A   20 CA F1     JSR 61898
49213   C03D   4C 00 C0     JMP 49152
49216   C040   60           RTS
```

ri in fondo al programma per poi riutilizzarli quando serve: in caso di utilizzo di numerosi indirizzi la tecnica è fondamentale. Dopo l'inizializzazione vengono chiamate due sottoprocedure della ROM, la SCNKEY ($EA87) che "legge" la tastiera, e la GETIN ($F13E) che riporta in A il valore ASCII del tasto premuto. Per i controlli si sottrae $30 (il numero "1" corrisponde al codice ASCII $31). A seconda sia stato premuto il tasto 1, 2 oppure 3, un diverso messaggio viene stampato sullo schermo; premendo il tasto 0 il programma termina. Per scrivere sullo schermo si sfrutta un'altra procedura della ROM, la CHROUT ($F1CA) che trasferisce sullo schermo il carattere il cui codice ASCII è contenuto in A. Quest'ultima parte sfrutta l'indirizzamento indiretto indicizzato (con X) per puntare al messaggio desiderato, che si può trovare ovunque nella memoria, e la cui lunghezza è indicata da Y.

Naturalmente il gruppo di programmi commentati vuole essere solo una base di partenza, una proposta che viene offerta all'utente allo scopo di escogitare nuove applicazioni e soluzioni sfruttando appieno la propria fantasia. Quindi, in bocca al lupo e... buon divertimento!

La tabella riepilogativa

Giunti a questo punto del corso sul Linguaggio Macchina, abbiamo pensato di riepilogare sia i concetti di cui ormai siamo padroni (!) sia quelli che conosciamo (forse solo per sentito dire), in una tabella che comprende e riguarda tutto il set di istruzioni del 6502, microprocessore utilizzato dai nostri beneamati Commodore 64, VIC 20 ed ora, nella versione 7501, anche dal Commodore 16 e dal PLUS 4.

La tabella è rivolta fondamentalmente a chi di Linguaggio Macchina conosce già qualcosa, non fosse altro che per il nostro corso. Teniamo a chiarire che può essere utile tanto per l'esperto, che non può ricordare a memoria proprio tutto (a meno che non sia un "maghetto"), quanto per il principiante, che il più delle volte non sa proprio dove sbattere la testa per trovare quello che gli serve (e ci siamo passati tutti!).

Dunque una guida pratica, comoda, che può essere utilizzata in qualsiasi situazione e per qualsiasi scopo (e in questo momento sto pensando al buon caro e vecchio "bigino"). Ma bando ai preamboli ed entriamo nel merito della tabella.

Il set completo di istruzioni utilizzabili dal programmatore in Linguaggio Macchina, consta di 56 operazioni diverse. Ciascuna di queste operazioni viene utilizzata sfruttando un determinato indirizzamento. In tutto sono disponibili 13 indirizzamenti, che analizzeremo più avanti. Naturalmente non sono permessi tutti i possibili abbinamenti tra operazione ed indirizzamento e, alla resa dei conti, potremo sbizzarrirci con "solo" 151 istruzioni.

A scanso di equivoci e di telefonate di protesta in redazione, premettiamo immediatamente che in questa tabella è stato volutamente tralasciato un dato caratteristico del funzionamento del microprocessore. Alludiamo al numero dei "cicli" necessari per ciascuna istruzione o, per dirla all'americana, al "timing".
Questa omissione è stata voluta proprio per motivi di chiarezza e di utilità: la tabella dovrà servire principalmente non a "mostri" di programmazione, ma ad utenti normali (senza offesa per alcuno, naturalmente) ai quali di timing e non timing interessa ben poco. Fra l'altro, detto fra noi, chi ha già utilizzato seriamente il computo dei cicli di un programma scagli pure la prima... telefonata!

La tabella sfrutta la caratteristica accennata in precedenza: ogni operazione interviene in un campo diverso, dando luogo alla vera e propria istruzione.

Verticalmente, nella prima colonna a partire da sinistra, sono riportate le parole chiave a nostra disposizione (ADC, AND, eccetera), mentre orizzontalmente, nella prima riga in alto, le varie indicizzazioni abbinabili.

Il punto di incontro della riga con la colonna interessata, individuerà un campo, caratteristico dell'istruzione che vogliamo utilizzare (non tacciateci di banalità, non siamo tutti geni...). Questo campo contiene in realtà tre valori diversi, che vanno sempre tenuti presenti al momento di programmare. Le prime due cifre riguardano il codice utilizzato per distinguere quella determinata istruzione. I numeri sono diversi, ma solo apparentemente, perchè l'unica differenza è costituita dalla loro base numerica: il primo è espresso in notazione decimale (#), mentre il secondo in quella esadecimale ($). Tale accorgimento può sembrare superfluo, ma spesse volte ci si può trovare a dover utilizzare una base piuttosto che l'altra, vuoi per un compilatore assembler non troppo sofisticato, vuoi perchè si sta creando un programma con delle semplici POKE da BASIC, e vi posso garantire che gli errori di conversione sono veramente grossolani e madornali.

Dunque, il primo valore esprime il codice operativo in notazione decimale, il secondo in esadecimale, ed il terzo riguarda la lunghezza effettiva

di ogni istruzione, cioè lo spazio utilizzato in memoria.

Ogni istruzione può richiedere 1, 2 oppure 3 byte a seconda dell'indirizzamento utilizzato. Il terzo valore del campo (N) riguarda appunto la sua lunghezza globale. E' importante ricordare che il numero riportato è comprensivo dell'istruzione stessa. Quando, per esempio, si dice di una istruzione che è "a due byte", vogliamo significare (e non solo noi) che oltre al codice operativo stesso abbiamo un solo byte di lavoro.

A questo punto, prima di procedere con la chiarificazione circa gli indirizzamenti, si rende necessario un esempio. Supponiamo di avere l'istruzione JSR $A183 e di volere sapere quali valori vengono effettivamente posti in memoria. Innanzitutto cerchiamo la parola chiave nella colonna verticale, poi procediamo orizzontalmente: l'unico indirizzamento disponibile è quello assoluto (guarda caso). Il campo che abbiamo individuato contiene 3 valori: 032, 20, 3.

Il primo, che per il momento non interessa, è espresso in decimale; il secondo è il codice operativo che cercavamo, mentre il terzo ci dice che la nostra istruzione richiede altri due bytes per specificare l'indirizzo di partenza della nostra subroutine. L'indirizzamento assoluto (vedi oltre) vuole che i due bytes successivi all'OP CODE siano espressi nella forma LB, HB. La nostra istruzione si presenta dunque nella memoria come: "20 83 A1". Se volessimo creare una routine in LM, partendo da Basic con i soliti READ, DATA e POKE, nelle istruzioni DATA dovremmo digitare i seguenti numeri decimali: "DATA 32,131,161". Chiarito il chiaribile, analizziamo in dettaglio i vari indirizzamenti.

Gli indirizzamenti

• IMPLICITO. Un' istruzione implicita deriva il suo nome dal fatto che non contiene specificamente l'indirizzo dell'operando su cui opera, ed infatti non richiede niente oltre al codice operativo stesso. Tipicamente viene utilizzato per operazioni che riguardano esclusivamente i registri interni.
Esempio: INX=E8

• ASSOLUTO. Abbiamo già visto nell'esempio precedente che tale indirizzamento richiede 3 byte, dove il primo è il codice operativo stesso, ed i restanti due costituiscono l'indirizzo a 16 bit che specifica l'operando. Tale valore, dal momento che il 6502 è un microprocessore ad 8 bit (tratta cioè solo valori fino a 255), viene codificato nella memoria secondo il consueto ordine LB, HB ,ottenuti grazie alla ancora più consueta formula HB=int (N/256) , LB=N-256*HB .
Esempio: LDA $A9B0 = AD B0 A9

• IMMEDIATO. In questo caso l'operando non è contenuto nella locazione puntata dall'istruzione, ma è costituito dal secondo byte stesso; naturalmente (per la limitazione degli 8 bit) sono impegnati in totale solo due byte.
Esempio: LDA #$00 = A9 00

• PAGINA ZERO. Per definizione sono impiegati soltanto due byte e questo è ben lungi dal costituire una limitazione; si può considerare come una variante dell'indirizzamento assoluto, in quanto in pratica il 6502 pone automaticamente a zero l'HB dell'operando, senza quindi che ci sia bisogno di specificarlo. Questa forma impiegata consente non solo uno spazio minore, ma anche, e soprattutto, una maggiore rapidità di esecuzione in virtù dell'ormai accennato timing. Va impiegato tutte le volte che si eseguono operazioni sui registri in pagina zero, che come sappiamo, sono molto importanti per il normale funzionamento del calcolatore.
Esempio: LDA $FB = A5 FB

• ACCUMULATORE. Riguarda solo alcune istruzioni che interessano l'accumulatore, quindi prevalentemente operazioni di scorrimento. E' simile all'implicito, dal quale si differenzia solo concettualmente.
Esempio: LSR A = 4A

• INDIRETTO INDICIZZATO. Sfrutta l'indice X e serve soprattutto per creare tabelle in pagina zero che possono essere scorse modificando il registro X.
Esempio: LDA($FB,X) = A1 FB

• INDICIZZATO INDIRETTO. Utilizza l'indice Y e si differenzia dal precedente non solamente per l'indice utilizzato. Serve, infatti, prevalentemente per gestire tabelle presenti in qualsiasi parte della memoria. Abbinato opportunamente al precedente offre caratteristiche di flessibilità veramente eccezionali.
Esempio: LDA ($FB),Y = B1 FB

• ASSOLUTO INDICIZZATO in X e Y. Nonostante sia molto più semplice del precedente, offre ugualmente ampie possibilità. Grazie a questo indirizzamento si può modificare l'operando di un'istruzione assoluta facendo variare un indice; naturalmente sono necessari 3 byte. Supponiamo di avere il seguente programmino:

```
LDX #$05
LDA $A000,X
RTS
```

L'accumulatore sarà caricato con il contenuto della locazione $(A000+05).
Esempio: LDA $A000,X = BD 00 A0

• PAGINA ZERO INDICIZZATO IN X ED IN Y. Variante del precedente, applica le caratteristiche di indicizzazione alle operazioni in pagina zero. Valgono tutte le indicazioni già esposte relative all'indirizzamento in pagina zero. Esiste una fondamentale differenza tra questo indirizzamento e quello indiretto indicizzato (e indicizzato indiretto), insita proprio nell'indirezione.
Esempio: LDA $A1,X = B5 A1

• INDIRETTO. Viene utilizzato solo da un'istruzione: JMP($XXXX). L'esecuzione del programma viene trasferita ad una locazione individuata sfruttando la stessa tecnica di indirezione di cui parlavamo prima. Esempio: se l'istruzione in questione è JMP ($A123), il registro $A123 contiene XX ed il registro $(A123+01) contiene YY, il contatore di programma verrà caricato con il valore $YYXX, e da qui proseguirà l'esecuzione.
Esempio: JMP ($A123) = 6C 23 A1

Conclusa la carrellata sugli indirizzamenti, scopriamo altre due colonne sulla tabella che sono importantissime per la programmazione e per

la comprensione del Linguaggio Macchina. Immediatamente a destra della colonna che riporta le parole chiave, viene fornita una breve descrizione, che aiuta a meglio comprendere il compito realmente svolto dall'istruzione.

Tali indicazioni sono sia in inglese che in italiano. Questa doppia nomenclatura può servire per meglio abbinare mnemonicamente il nome con l'operazione eseguita. Per esempio, pensando a "BEQ", viene più spontaneo collegarlo a "Branch on EQual" che a "salta se uguale", ma comunque è solo questione di abitudine.

Oltre a ciò, vengono indicati, nell'ultima colonna a destra, tutti i singoli bit che compongono il byte di status, fondamentale per il funzionamento del calcolatore. Nell'ordine sono riportati:

I-Interrupt N-Negative
D-Decimal Z-Zero
V-oVerflow C-Carry

I bit interessati dalla determinata istruzione sono segnati con un asterisco, con uno 0 oppure con un 1 se lo scopo dell'istruzione è espressamente quello di modificarli.

In fondo alla tabella è stata aggiunta una legenda molto schematica per la corretta interpretazione di tutti i simboli, convenzionali e non, utilizzati. Ultima nota di ordine tipografico è quella di prestare attenzione alla somiglianza fra la lettera "O" la lettera "D" ed il numero "0": questo non è stato sbarrato diagonalmente, ma la differenza è veramente notevole se si osserva con attenzione.

Bene, crediamo di avere parlato fin troppo, quindi vi lasciamo alla tabella, trampolino di lancio verso l'affascinante mondo del Linguaggio Macchina.

Gli indirizzamenti come li vede il computer

Come conclusione di questo minicorso di programmazione in linguaggio macchina, abbiamo ritenuto utile riportare alcune tabelle per chiarire meglio il procedimento che segue il computer per interpretare le istruzioni. Ben lontano dal voler mostrare i flussi di dati all'interno del microprocessore, i grafici qui riportati illustrano la condizione dei registri A, X e Y, sia prima che dopo una istruzione.

Lo scopo, molto evidente, è quello di familarizzare con la logica del calcolatore, dal momento che il vero segreto per programmare in linguaggio macchina, è di riuscire a pensare allo steso modo del computer. Non diversamente che per lo studio delle lingue, se si vogliono ottenere buoni risultati in una conversazione, la difficoltà, una volta imparata la sintassi ed un minimo di vocabolario, consiste nel pensare in quella determinata lingua, e non certo a prodigarsi come improvvisati traduttori (per quello scopo esistono appositi programmi, pardon, persone!).

Un buon sistema per avvicinarsi al linguaggio macchina consiste certamente nello studio approfondito di queste tabelle, una per ciascun tipo di indirizzamento possibile, a cui naturalmente deve seguire un debito esercizio di modifica, ampliamento, fino ad arrivare al... primo programma !

La consultazione dei grafici non comporta pàrticolari difficoltà, ma è bene dare loro uno sguardo insieme.

• Il primo "campo" in alto rappresenta i registri A, X e Y prima dell'esecuzione dell'istruzione; se vi sono riportati alcuni valori, questi sono chiamati in causa nell'esecuzione dell'istruzione; altrimenti, se sono vuoti, non vuol dire che saranno posti a zero, ma solamente che il dato in essi contenuto non entra in ballo nella specifica istruzione.

• Il secondo campo rappresenta la pagina zero, ovvero il segmento di memoria numerato da $00 a $FF; in questa "pagina" di memoria sono contenuti importanti puntatori e locazioni, utilizzate dal Sistema Operativo, che in questa sede non interessano.

E' importante sapere invece che alcune locazioni (da $FB a $FE comprese), utilizzabili dall'utente, sono di grandissima utilità per determinate istruzioni. Naturalmente, per motivi di spazio, non abbiamo rappresentato tutti i 256 byte che compongono la pagina zero, ma solo uno "spaccato" di essa, in cui sono indicati l'inizio ($00) e la fine ($FF); anche in questo caso le locazioni vuote indicano semplicemente che non sono utilizzate per questi esempi e possono contenere un qualsiasi valore.

• Il terzo campo, contrassegnato genericamente con il termine "memoria", in realtà rappresenta solo un banco di 4K (4096 byte) il cui uso è riservato appositamente per i microprogrammi in linguaggio macchina proposti nella stessa figura.

La zona di memoria interessata non è alterabile da Basic (se non ricorrendo a POKE) e resiste anche al system reset (SYS 64738): i dati non si cancellano fino allo spegnimento del computer. In realtà la memoria disponibile parte da $0801 e va fino a $9FFF, oltre a questo banco da $C000 a $CFFF, ma, data la presenza del Basic, si rischia di cancellere le nostre preziose routine per un nonnulla, a meno di spostare i puntatori di inizio o fine del Basic... troppo complicato.

Alcuni problemi si possono incontrare quando si utilizza un programma di Monitor che risiede proprio (guarda caso) in quei 4K a $C000: un buon compromesso può essere l'utilizzo delle locazioni a partire da $8000 (#32768), ma attenzione a non dimensionare vettori troppo lunghi !

Anche in questo caso abbiamo rappresentato solo alcuni dei 4096 bytes, evidenziando i primi e gli ultimi. A partire proprio da $C000 (#49152) sono allocate le istruzioni degli esempi, per cui, dopo avere caricato, con delle POKE o con programmi monitor, tutti i registri e le locazioni usate con i valori previsti, digitando ⟨SYS 49152⟩ e verificando i registri, otterremo i risultati voluti. Come al solito, le locazioni vuote sono ininfluenti ai fini della tabella.

• Nel quarto campo sono finalmente contenuti i registri A, X e Y dopo l'esecuzione dell'istruzione: rappresentando la situazione terminale dell'istruzione stessa, servono per farci capire l'operato di ciascun indirizzamento. Ormai dovrebbe essere chiaro, ma è bene ricordarlo, che sono riportati solo i valori dei registri interessati.

A questo punto non resta che analizzare ad una ad una le 11 tabelle relative agli indirizzamenti.

Indirizzamento

Implicito
TAX [AA]

PRIMA ...

A	C2
X	
Y	

PAGINA ZERO

ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	
FC	
FD	
FE	
FF	

MEMORIA

CØØØ	AA
CØØ1	6Ø (RTS)
CØØ2	
CØØ3	
CØØ4	
CFFB	
CFFC	
CFFD	
CFFE	
CFFF	

... E DOPO

A	C2
X	C2
Y	

Immediato
LDA #$13 [A9 13]

PRIMA ...

A	
X	
Y	

PAGINA ZERO

ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	
FC	
FD	
FE	
FF	

MEMORIA

CØØØ	A9
CØØ1	13
CØØ2	6Ø (RTS)
CØØ3	
CØØ4	
CFFB	
CFFC	
CFFD	
CFFE	
CFFF	

... E DOPO

A	13
X	
Y	

Assoluto
LDX $CFFD [AE FD CF]

PRIMA ...

A	
X	
Y	

PAGINA ZERO

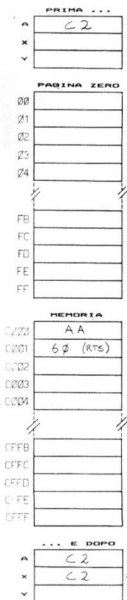
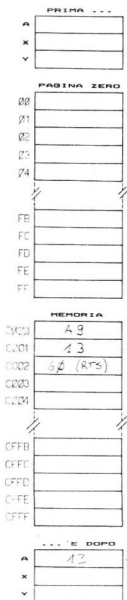
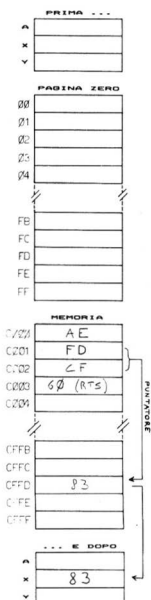
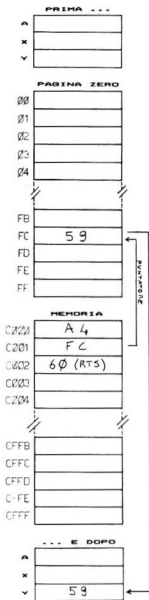
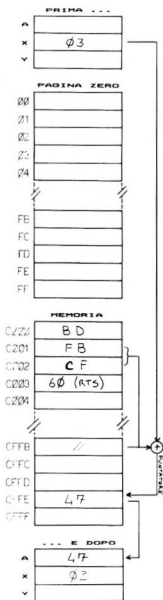
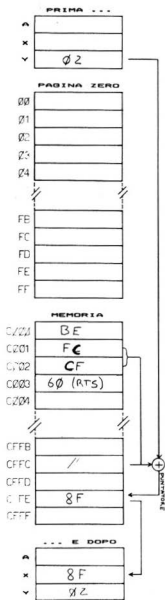
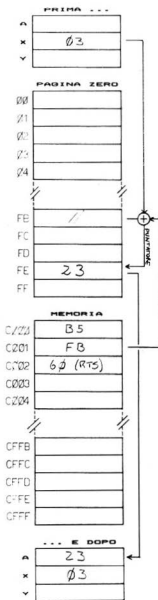
ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	
FC	
FD	
FE	
FF	

MEMORIA

CØØØ	AE
CØØ1	FD
CØØ2	CF
CØØ3	6Ø (RTS)
CØØ4	
CFFB	
CFFC	
CFFD	83
CFFE	
CFFF	

PUNTATORE

... E DOPO

A	
X	83
Y	

Pagina zero
LDY $FC [A4 FC]

Assoluto, X
LDA $CFFB,X [BD FB CF]

Assoluto, Y
LDX $CFFC,Y [BE FC CF]

Pag. zero, X
LDA $FB,X [B5 FB]

Pag. zero, Y
LDX $FC,Y [B6 FC]

Accumulatore
ASL [0A]

PRIMA ...

A	
X	Ø3
Y	

PAGINA ZERO

ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	//
FC	
FD	
FE	23
FF	

PUNTATORE

MEMORIA

CØØØ	B5
CØØ1	FB
CØØ2	6Ø (RTS)
CØØ3	
CØØ4	
CFFB	
CFFC	
CFFD	
CFFE	
CFFF	

... E DOPO

A	23
X	Ø3
Y	

PRIMA ...

A	
X	
Y	Ø2

PAGINA ZERO

ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	
FC	//
FD	
FE	33
FF	

PUNTATORE

MEMORIA

CØØØ	B6
CØØ1	FC
CØØ2	6Ø (RTS)
CØØ3	
CØØ4	
CFFB	
CFFC	
CFFD	
CFFE	
CFFF	

... E DOPO

A	
X	33
Y	Ø2

PRIMA ...

A	Ø4
X	
Y	

= % 0000 0100

PAGINA ZERO

ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	
FC	
FD	
FE	
FF	

MEMORIA

CØØØ	ØA
CØØ1	6Ø (RTS)
CØØ2	
CØØ3	
CØØ4	
CFFB	
CFFC	
CFFD	
CFFE	
CFFF	

... E DOPO

A	Ø8
X	
Y	

= % 0000 1000

Indiretto indic., X
LDA ($02,X) [A1 02]

PRIMA ...

A	
X	FB
Y	

PAGINA ZERO

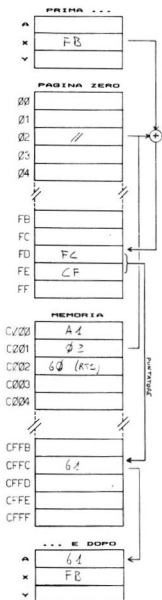
ØØ	
Ø1	
Ø2	//
Ø3	
Ø4	
FB	
FC	
FD	FC
FE	CF
FF	

MEMORIA

CØØØ	A1
CØØ1	Ø2
CØØ2	6Ø (RTS)
CØØ3	
CØØ4	
CFFB	
CFFC	61
CFFD	
CFFE	
CFFF	

PUNTATORE

... E DOPO

A	61
X	FB
Y	

Indic. indiretto, Y
LDA ($FB),Y [B1 FB]

PRIMA ...

A	
X	
Y	Ø2

PAGINA ZERO

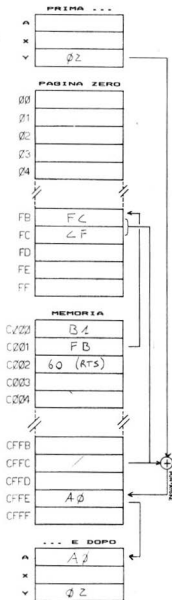
ØØ	
Ø1	
Ø2	
Ø3	
Ø4	
FB	FC
FC	CF
FD	
FE	
FF	

MEMORIA

CØØØ	B1
CØØ1	FB
CØØ2	60 (RTS)
CØØ3	
CØØ4	
CFFB	
CFFC	/
CFFD	
CFFE	AØ
CFFF	

PUNTATORE

... E DOPO

A	AØ
X	
Y	Ø2

Le routine grafiche di Danilo Toma

PRESENTAZIONE

Molti di voi, credo, avranno già sentito parlare delle routine grafiche pubblicate sul N.14 di Commodore Computer Club e magari avranno maledetto il mio nome quando venivano frustrati da errori di copiatura dell'immenso listato; prova ne sono le numerose e accorate telefonate ricevute da quando uscì in edicola quel numero della rivista.

In un paio di occasioni il mio telefono ha squillato minaccioso anche dopo la mezzanotte e mi sono sentito domandare se non avevo di meglio da fare che perseguitare poveri hobbisti proponendo loro listati così ponderosi: a tale punto arrivava la disperazione di qualche lettore.

Voglio perciò in questa occasione, in cui vengono ripresentate le routine (integrate con altri utili comandi) illustrare le peripezie che mi hanno portato alla loro creazione così da indurvi in una maggiore condiscendenza nei miei confronti.

La storia

Era una notte buia e tempestosa quando portai alla Systems un Editor di sprite che avevo realizzato e che de Simone trovò interessante. Gli consegnai allora anche una routine in linguaggio macchina che plottava punti nella pagina grafica. Il Grande Capo (!) però, preso da innumerevoli impegni, non trovava il tempo di esaminarla: così, per allettarlo maggiormente, realizzai un'estensione di tale routine che consentiva di tracciare linee oltre che punti.

Fu un lavoro tremendo di messa a punto, dal momento che non disponevo nemmeno di un assemblatore: facevo tutto "a mano" e se ci penso mi vengono i brividi. Alla fine però riuscii nell'intento e consegnai la nuova "release" che fu subito pubblicata sul n. 10 di C.C.C.

I lettori di più lunga data ricorderanno ancora le speciali variabili intere, a cui assegnare i valori delle coordinate, che caratterizzavano quella versione.

Pungolato dal successo ottenuto e, minacciato da de Simone volli migliorare ancora il programma e per non sottrarre troppo tempo agli studi universitari decisi di portarmi il computer in vacanza.

Fu una decisione tragica: mettersi davanti al computer in agosto, dentro una roulotte cotta dal sole, è il massimo del masochismo. Inoltre la mia ragazza non gradiva starsene da sola sulla spiaggia mentre io digitavo incomprensibili numeri sull'amata-odiata tastiera e quindi fantozziani erano all'ordine del giorno. Ero giunto al punto di desiderare il brutto tempo per avere una scusa plausibile per non uscire dalla caravan di mio fratello.

Alla fine riuscii a terminare il progetto ma promisi a me stesso che non avrei mai più ripetuto una simile esperienza: dall'estate seguente avrei consacrato le vacanze al riposo portando solo un piccolo Ms/Dos....

Il resto della storia è noto: nel n.14 di Commodore Computer Club venne pubblicato quel sofferto lavoro e... ricominciarono le sofferenze, questa volta telefoniche e da parte dei lettori.

Da aggiungere solamente un'ultima nota rivolta agli esperti di linguaggio macchina che esamineranno i disassemblati commentati: riguardando le routine mi sono accorto di alcune ingenuità, alcuni problemi che avrei potuto risolvere in modo più "brillante". Siate comprensivi, era una calda estate...

Danilo Toma

Premessa

Il criterio seguito nella stesura di queste pagine è quello di consentire, a chi già dispone del nucleo centrale delle routine grafiche (nella versione pubblicata sul n. 14 di C.C.C.), l'inserimento, senza difficoltà, delle istruzioni di cui non dispone. Per tale motivo si è preferito mantenere distinti i vari "moduli" del programma anziché riunire i nuovi quindici comandi in un unico listato.

Chi, invece, non sa nulla delle famigerate routine, non deve fare altro che seguire con attenzione i paragrafi che seguono.

Introduzione alla grafica del C/64

Prima di parlare delle routine qui presentate, è opportuno introdurre alcune nozioni riguardo alla grafica del Commodore 64 poiché sul manuale in dotazione non viene spiegato nulla a tale riguardo, tranne una breve descrizione sul funzionamento degli sprite.

Consigliamo, a chi volesse approfondire, alcuni degli articoli sulla grafica apparsi su Commodore Computer Club:

- N.28: "La tua prima volta con la grafica del C/64"
- N.30: "Tutti i caratteri del C/64"
- N.32: "La gestione dei quattro banchi di memoria"
- N.35: "Tutto sugli Sprite"

Riassumiamo brevemente, comunque, il funzionamento del C/64 in ambiente grafico.

Quella che normalmente compare sullo schermo, la cosiddetta "Pagina Testo", non è altro che la visualizzazione di 1000 byte della memoria (dalla locazione 1024 alla 2023). Oltre a questa il C/64 ha la possibilità di mostrare, agendo su opportune locazioni di memoria, una pagina (o mappa) grafica.

Anche in questo caso viene visualizzata una parte della memoria, ma non si tratta più di 1000 byte bensì di 8000! E poiché ogni byte è formato da 8 bit, il calcolo conduce al risultato di 64000 bit, ai quali corrispondono sullo schermo altrettanti punti (nel modo standard) disposti in una matrice di dimensioni 320x200.

Ogni singolo bit può essere posto, indipendentemente dagli altri, a "1" oppure a "0", determinando l'accensione o lo spegnimento del corrispondente punto sullo schermo. Purtoppo la corrispondenza bit/punto è piuttosto complicata: se definiamo con "I" la locazione inziale della pagina grafica e con "X" e "Y" le coordinate del punto scelto, l'origine degli assi è posta in alto a sinistra; per settare il punto, è dunque necessario eseguire:

```
L=I+320*INT(Y/8)+8*INT(X/8)+
YAND7
:B=7-(XAND7):
POKEL,PEEK(L)OR2#B
```

La pagina grafica utilizzata nell'ambiente Toma si estende dalla locazione 57344 alla locazione 65343, sfruttando gli 8K di RAM "celati" dal Sistema Operativo (Kernal). Ciò comporta qualche piccola complicazione (di cui si parlerà più avanti) ma presenta il notevole vantaggio di non sottrarre spazio ai programmi.

Il colore dei punti viene determinato, nel modo grafico standard, dalle 1000 locazioni della memoria di schermo (che, a tale scopo, viene spostata alle locazioni 52224-53223).

Ogni locazione controlla il colore di un quadratino di 8x8 punti: i 4 bit più bassi della locazione determinano il colore dei punti spenti (lo sfondo), i 4 più alti il colore dei punti accesi.

Ad esempio:

```
POKE52224,16*A+B
```

determina i colori dei primi 8x8 punti in alto a sinistra, mentre:

```
POKE53223,16*A+B
```

i colori degli ultimi in basso a destra. Saranno di colore "A" i punti accesi e di colore "B" quelli spenti. Potete quindi scegliere, per ogni singolo punto, tra due soli colori ma potete cambiare a volontà tali colori per ciascun blocco di 8*8 punti.

Oltre al modo grafico standard esiste un'altra configurazione della pagina grafica (il modo multicolore) che permette di scegliere, per ogni punto, tra quattro colori anziché due.

Questa più ampia scelta va a discapito della risoluzione orizzontale che passa da 320 a 160 punti (la metà). In pratica ogni punto diventa largo il doppio.

Vediamo perché. La macchina, per sapere quale colore usare nel visualizzare un punto sullo schermo, controlla il bit corrispondente e agisce in base al valore di questo ma, come tutti i lettori dovrebbero sapere, un bit può assumere due soli valori: 0 oppure 1. Per avere quattro valori (e quindi quattro colori) bisogna prendere in considerazione una coppia di bit. Tale generica coppia può presentarsi in quattro configurazioni:

Coppia tipo 0:00
Coppia tipo 1:01
Coppia tipo 2:10
Coppia tipo 3:11

Badate che questi sono numeri binari e corrispondono ai decimali 0; 1; 2; 3, a cui d'ora in poi sarà fatto riferimento per comodità.

Col metodo descritto, dunque, i punti sono larghi il doppio dei punti "normali". Si viene allora a determinare una sovrapposizione-coincidenza tra ogni punto di ascissa pari (2*n) e il punto di ascissa dispari (2*n+1) che lo segue.

Ad esempio i punti di coordinate (50,20) e (51,20) in modo Multicolore vengono considerati identici.

Le informazioni sui colori da usare provengono, nel modo Multicolore, da tre fonti diverse: dalla locazione 53281 quando la coppia di bit è la 0, dai quattro bit più alti della memoria di schermo (locazioni 52224-53223) per le coppie 1, dai quattro bit più bassi per le coppie 2, dalla memoria colore (locazioni 55296-56295) per le coppie 3.

Vale lo stesso discorso del modo Standard per il cambio dei colori ogni 8*8 punti, escluso il colore delle coppie 0 (lo sfondo) che vale per l'intero schermo.

La terza dimensione

Certamente avrete tutti visto una fotografia e, osservandola, avrete ricevuto l'impressione della cosiddetta "profondità".

Alcuni soggetti sembrano più lontani ed altri più vicini. Nella fotografia, in altre parole, un'immagine reale, tridimensionale, è stata ridotta ad un'immagine bidimensionale ma ha ugualmente conservato l'illusione della profondità.

Si può ottenere lo stesso effetto sullo schermo grazie ad alcune formule matematiche, ed è esattamente ciò che simulano le routine di Toma.

Basta assegnare le solite coordinate X e Y e in più aggiungere l'illusoria distanza dallo schermo, Z, a cui si trova il punto desiderato.

Per familiarizzare con la nuova coordinata vi consigliamo di copiare, appena digitate le routine grafiche (e solo allora!) i semplici "demo" presenti in queste pagine.

Comunque, se proprio non vi ci raccapezzate, basta assegnare sempre valore nullo alla coordinata Z per eliminare l'effetto prospettico.

"padrone di casa" a parte una sola eccezione da imputare, forse, ad una svista degli autori del Basic del C/64: dopo l'istruzione THEN eventuali nuovi comandi introdotti dall'utente non vengono riconosciuti. Perchè ciò non accada, basta frapporre il carattere di doppio punto (:) tra THEN e il nuovo comando.

Le nuove istruzioni vanno precedute dal carattere di "freccia a sinistra", che si ottiene premendo il tasto situato in alto a sinistra della tastiera.

Per abilitare una nuova routine di interpretazione basta eseguire SYS 51000 (ovviamente dopo aver caricato in memoria le routine).

V'è anche una nuova routine di gestione degli errori che ha lo scopo di riportare il video al modo testo ogni volta che si verifica un errore oppure finisce un programma. Se volete che rimanga visualizzata la pagina grafica fate terminare il programma con l'istruzione STOP oppure premete il tasto Run/Stop.

In tal caso, per tornare alla pagina testo il modo più semplice è premere i due punti (:) e il tasto Return.

Caratteristiche generali

Copiate con pazienza il primo programma (dal nome "Routine Grafiche") con tutta l'attenzione e la pazienza necessarie. Molti accorgimenti sono stati introdotti, rispetto alla versione del N.14 di C.C.C. per limitare errori di digitazione. Prima di dare il RUN eseguite, ovviamente, una copia di sicurezza su disco o nastro: in caso di malfunzionamenti non sarete costretti a ridigitare tutto daccapo.

I nuovi comandi non danno alcun problema di funzionamento se usati correttamente e sono ben tollerati dal

Tale routine viene abilitata con:

SYS 49724

Vi ricordiamo che un eventuale software reset (SYS 64738) non cancella assolutamente tali routine ma le disabilita soltanto. Non risulta dunque necessario ricaricarle da nastro o disco.

L'intervallo dei parametri

I valori che si possono assegnare all'ascissa e all'ordinata sono compresi tra il numero negativo -32768 e il positivo +32767. Per la coordinata Z (la profondità) e per i raggi delle istruzioni CIRCLE e ARC l'intervallo valido è tra 0 e +32767. Al di fuori dei limiti consentiti il programma si blocca con l'emissione di un "Illegal quantity error".

La lunghezza massima di una linea non può eccedere i 32767 punti, altrimenti si verifica un overflow nei calcoli e la linea risultante sarà "sballata".

L'origine degli assi, cioè il punto di coordinate (0,0,0), è posto al centro dello schermo.

Vengono individuati con ordinata (Y) positiva i punti al di sopra dell'origine e con ascissa (X) positiva quelli alla sua destra.

Nel caso in cui una figura da disegnare ecceda i bordi dello schermo, ne viene tracciata la sola parte visibile mentre, per i punti al di fuori, viene inibito il plottaggio (che invaderebbe aree di memoria pericolose).

Il controllo di tracciabilità viene quindi effettuato su ogni singolo punto, e, se in termini di velocità ciò non viene notato con linee di 300-400 punti, risulta evidente quando, per una linea da 30000 punti, occorrono circa 5 secondi per completare l'esecuzione.

Premere il tasto Restore mentre il computer sta eseguendo un'operazione di tracciamento comporta l'inchiodamento della macchina, se non avete aggiunto i pochi byte necessari a neutralizzare tale tasto (vedi dopo il paragrafo "Disabilita tasto restore"). Pertanto, se volete fermare un programmma, usate solo il tasto Run/Stop e, in seguito, il carattere di doppio punto (:) e il tasto Return.

Durante il tracciamento, inoltre, l'orologio interno (quello che potete leggere con la variabile TI$) non viene incrementato.

Questi difetti sono dovuti al fatto che, per non rubare memoria ai programmi in Basic, la pagina grafica (di 8K Ram) viene sistemata "sotto" il Sistema Operativo (dalla locazione 57344 alla 65343) e questa "convivenza" genera i piccoli inconvenienti citati.

Un ultimo avvertimento: la pressione contemporanea di Run/Stop e Restore "sporca" la parte bassa della pagina grafica.

Opzione $

Per eseguire più velocemente le istruzioni di tracciamento (PLOT, DRAW, ARC, CIRCLE) si possono, in alcuni casi, sostituire i parametri di tali comandi con il simbolo del dollaro ($).

I parametri delle istruzioni citate, dopo essere stati valutati, sono conservati in una zona di memoria da dove vengono successivamente prelevati e utilizzati per l'esecuzione del comando.

Se uno di questi parametri viene rimpiazzato con il simbolo "$", allora le routine "saltano" la valutazione e il parametro viene conservato; quando in seguito, nel prosieguo dell'esecuzione, la routine preleva i valori dal "magazzino", trova il valore che era stato depositato con il comando precedente.

Ad esempio se digitate (facendo, ovviamente precedere i comandi dal carattere di freccia a sinistra):

COLOR1: PLOT10,5,0
COLOR0: PLOT$,$,$

vedrete che il punto di coordinate (10,5,0) verrà prima acceso e poi spento.

L'uso dell' opzione "$" permette risparmi di tempo tanto maggiori quanto più complesse sono le espressioni che dovrebbero, in caso contrario, essere eseguite.

Attenzione che le locazioni del "magazzino" vengono utilizzate dalle quattro istruzioni con alcune sovrapposizioni; quindi, per evitare che l'output grafico sia diverso da quello previsto, usate quest'opzione solo se l'ultimo comando grafico (PLOT, DRAW, ARC, CIRCLE) era dello stesso tipo di quello con i dollari ($).

Ad esempio:

CIRCLE0,0,0,50,50
PLOT20,10,0
CIRCLE$,$,$,80,80

In quest'ultimo caso il secondo cerchio avrà come coordinate del centro quelle di PLOT e non quelle del cerchio precedentemente tracciato.

Sprite

L'utilizzo degli sprite in pagina grafica richiede alcuni accorgimenti.

Il processore video, infatti, gestisce solamente 16K di memoria e, quando è abilitato il modo grafico con GRAF oppure MGRAF, tali 16K partono dalla locazione 49152.

Di conseguenza i dati degli sprite dovrebbero essere memorizzati a partire da tale locazione; da questa, però, sono allocate anche le istruzioni l.m. delle stesse routine grafiche!

Inoltre le locazioni che indicano dove si trovano i dati di ogni sprite non sono più allocate in 2040-2047 ma in 53240/53247.

Per il resto (colori, coordinate, collisioni, eccetera) le locazioni di riferimento rimangono invariate.

Ad esempio, se vogliamo disporre dello sprite 0 in pagina grafica, possiamo memorizzare i dati a partire dalla locazione 51712, che corrisponde a 49152+40*64, quindi nella locazione 53240 metteremo il valore 40.

Attenzione, comunque, a non invadere zone occupate dalle routine grafiche!

Altro avvertimento: le locazioni 53240/53247 vengono resettate ogni volta che vengono eseguiti GRAF e MGRAF quindi dopo tali istruzioni vanno "ri-pokati" i valori in tal modo cancellati.

Le prime nove istruzioni

Dopo aver digitato il primo blocco (programma "Routine Grafiche"), avete a disposizione nove istruzioni grafiche che, se avete digitato correttamente il programma, hanno una sintassi ben precisa.

N.B. Tutti i comandi devono esser preceduti dal carattere di freccia a sinistra.

CLEAR

Sintassi: CLEAR
Funzione: pulisce la pagina grafica (cioè pone tutti i bit a 0)

GRAF

sint.: GRAF A,B
funz.: fa passare al modo grafico Standard settando il colore A per lo sfondo e il colore B per i punti accesi.

"A" e "B" rappresentano i codici dei colori e possono assumere valori compresi tra 0 e 15.

MGRAF

sint.: MGRAF A,B,C,D
funz.: fa passare al modo grafico Multicolore settando il colore A per le coppie di bit 0 (lo sfondo), il colore B per le coppie 1, il colore C per le coppie 2, il colore D per le coppie 3.

TEXT

sint.: TEXT A,B
funz.: fa passare al modo testo settando il colore A per lo sfondo e il colore B per i caratteri.

Tutti i caratteri presenti sullo schermo diventano del colore B.

COLOR

sint.: COLOR N
funz.: determina con quale dei colori, definiti con le istruzioni GRAF o MGRAF, vanno tracciati i punti. N può variare tra 0 e 3. Il valore N vale:

0 per punti del col. A
1 " " " " B
2 " " " " C
3 " " " " D

PLOT

sint.: PLOT X,Y,Z
funz.: setta il punto di coordinate (X,Y,Z). Il colore usato è quello specificato dall'ultimo comando COLOR.

X è la distanza orizzontale, misurata in punti-schermo, dal centro dello schermo (X assume valori positivi per i punti alla destra del centro).

Y è la distanza verticale dal centro (assume valori positivi nella parte superiore dello schermo)

Z è la distanza del punto dallo schermo (profondità apparente). Può assumere solo valori non negativi.

DRAW

sint.: DRAW X1,Y1,Z1,X2,Y2,Z2
funz.: traccia una linea di estremi (X1,Y1,Z1) e (X2,Y2,Z2). Il colore usato è quello specificato dall'ultimo comando COLOR

CIRCLE

sint.: CIRCLE X,Y,Z,RX,RY

funz.: traccia un'ellisse di centro (X,Y,Z). RX è Il semiasse orizzontale, RY quello verticale.

La circonferenza è un caso particolare dell' ellisse e si ottiene quando i due semiassi sono uguali (cioè RX=RY).

Il risultato "estetico", non certo esaltante di questo comando, è dovuto alla limitata risoluzione grafica del calcolatore.

ARC
sint.: ARC X,Y,Z,RX,RY,AI,AF,P
funz.: traccia archi d'ellisse (o di circonferenza) con centro (X,Y,Z) e semiassi RX,RY. AI è l'angolo iniziale, da cui fare partire il tracciamento, AF è l'angolo finale. Tali angoli vanno espressi, come per le altre funzioni trigonometriche, in radianti.

Prima di parlare dell'ultimo parametro di ARC (cioè P) e delle grandi possibilità che offre, è necessario spiegare come lavora la routine che esegue CIRCLE e ARC.

La risoluzione grafica del C/64, pur essendo già buona, è largamente insufficiente a rappresentare linee curve in modo "gradevole". Perciò, fatti i debiti confronti, ci si è basati sulla constatazione che, con un poligono di un numero rilevante di lati (quaranta o più), si ottiene un effetto visivo che ha pochissimo da invidiare ad una circonferenza tracciata punto per punto, con il notevole vantaggio di una maggiore rapidità di esecuzione.

E dal momento che già col metodo "poligonale" un'ellisse (o una circonferenza) di media grandezza viene tracciata in circa tre secondi, capirete che non vi sono stati dubbi nella scelta del sistema.

Il tempo interminabile di tracciamento è dovuto soprattutto al calcolo dei punti dell'ellisse, che richiede l'uso delle funzioni seno e coseno. Per tali calcoli ci si è limitati ad utilizzare le routine originali del Sistema Operativo che, per come sono strutturate, non risultano molto veloci.

E veniamo al parametro "P". Questo indica ogni quanti radianti vanno calcolati i punti dell'ellisse che devono essere uniti con linee rette. P dunque diminuisce all'aumentare del numero di lati del "poligono-ellisse". Nel caso dell'istruzione CIRCLE tale valore è fisso ed è circa uguale a 0.157; per essere più precisi è uguale a 2*pigreco/40, quindi le ellissi così ottenute sono in realtà poligoni di 40 lati.

Nel comando ARC, invece, "P" lo stabilisce l'utente. Ciò significa che oltre ai soliti archi potete fare disegnare poligoni con un numero qualunque di lati! E non solo poligoni, come si potrà notare facendo girare il programma dimostrativo dal nome "Poligoni Stelle".

Consiglio

Dopo aver digitato, registrato, verificato e lanciato (RUN) il primo blocco di programma ("Routine grafiche"), potete digitare e far girare i programmi dimostrativi riportati fino al paragrafo "Neutralizzazione del tasto Restore". Se avete digitato correttamente le routine, dovrete notare la visualizzazione delle immagini riportate in queste pagine. In caso contrario armatevi di pazienza e controllate riga per riga il listato: avete sicuramente sbagliato a digitare!

Grafici di funzione

Il semplice programma "Grafici di funzione" permette di visualizzare una qualsiasi funzione a patto di conoscere, naturalmente, l'analisi matematica.

Dopo il RUN comparirà un "menu" con due possibili scelte: cambio d'intervallo e cambio di funzione.

Premendo il tasto 2 verrà visualizzata una scritta con le istruzioni da seguire e, al di sotto, verrà listata la linea 240 che contiene la funzione. Nel programma riportato la funzione è:

240 DEF FNY(X)=SIN(X)

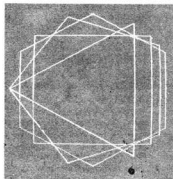
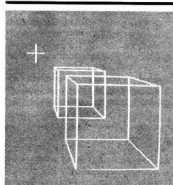
Premendo il tasto 1, invece, verranno chiesti gli estremi dell'intervallo in cui si desidera studiare la funzione. L'ampiezza dell'intervallo può variare a piacimento, così da constatare il comportamento delle funzioni più (intervallo ristretto) o meno (intervallo ampio) in dettaglio. Provate ad inserire i valori: -3.14,+3.14.

Dopo la pressione del tasto Return il programma calcolerà il massimo e il minimo della funzione nell'intervallo impostato. Si precisa che i due estremi, non essendo calcolati con il metodo delle derivate, non risultano molto precisi.

Passati alcuni secondi in paziente attesa compariranno i due dati e, al di sotto, un'altra richiesta.

Dovete comunicare al programma il fattore di moltiplicazione dei valori della funzione. Cioè dovete rispondere 1 per visualizzare la funzione in modo "normale". Rispondendo con numeri compresi tra 0 e 1 la funzione risulterà "schiacciata", mentre con numeri maggiori di 1 risulterà espansa in altezza. Rispondendo con 0, la funzione occuperà in altezza l'intero schermo.

Dopo quest'ultima risposta verranno tracciati gli assi, se visibili, e, finalmente, la funzione.

Per tornare al menu si preme un tasto qualunque.

Il listato

```
100 REM ********************
110 REM *                  *
120 REM * ROUTINE GRAFICHE *
130 REM * PER COMMODORE 64 *
140 REM * DI DANILO  TOMA  *
150 REM *                  *
160 REM ********************
170 :
180 PRINTCHR$(147);SPC(252)"LET
    TURA DEI DATI"
190 :
200 FOR K=1 TO 11:READ C,D,S:B=
    0
210 FOR I=C TO D:READ A:B=B+A:P
    OKE I,A:NEXT
220 IF B<>S THEN PRINT"ERRORE N
    EL BLOCCO N.";K:END
230 NEXT
240 :
250 SYS51000:SYS49724
260 PRINTCHR$(147);SPC(252)"COM
    ANDI ATTIVATI"
 280 :
 290 :
1000 REM   *****   BLOCCO 1    ***
     **
1010 :
1020 DATA  49152,49351,28136
1030 DATA  73,255,149,250,181,24
     9,73,255,149,249
1040 DATA  246,249,208,2,246,250
     ,160,202,162,255
1050 DATA  96,162,3,189,214,2,14
     9,93,202,16
1060 DATA  248,173,208,2,133,87,
     173,209,2,133
1070 DATA  88,173,210,2,133,91,1
     33,78,173,211
1080 DATA  2,133,92,133,79,96,23
     4,234,192,48
1090 DATA  12,3,0,85,170,255,234
     ,234,234,234
1100 DATA  234,234,160,232,162,0
     ,56,165,93,229
1110 DATA  87,133,249,165,94,229
     ,88,133,250,16
1120 DATA  3,32,0,192,140,55,193
     ,142,59,193
1130 DATA  160,198,138,208,2,160
     ,230,140,62,193
1140 DATA  160,232,162,0,56,165,
     95,229,91,133
1150 DATA  253,165,96,229,92,133
     ,254,16,5,162
1160 DATA  4,32,0,192,140,106,19
     3,142,110,193
1170 DATA  134,89,134,90,160,198
     ,138,208,2,160
1180 DATA  230,140,113,193,160,5
     6,169,229,224,0
1190 DATA  208,4,160,24,169,101,
     140,72,193,141
1200 DATA  75,193,141,81,193,141
     ,87,193,141,93
1210 DATA  193,169,234,141,115,1
     93,165,249,208,2
1220 DATA  165,250,208,27,169,96
     ,141,115,193,24
1230 :
1240 REM   *****   BLOCCO 2    ***
     **
1250 :
1260 DATA  49352,49551,30185
1270 DATA  144,3,32,104,193,32,1
     41,193,165,78
1280 DATA  197,95,208,244,165,79
     ,197,96,208,248
1290 DATA  96,230,249,208,2,230,
     250,230,253,208
1300 DATA  2,230,254,169,0,133,2
     51,133,252,133
1310 DATA  247,133,248,162,33,16
     5,247,56,229,249
1320 DATA  168,165,248,229,250,1
     44,5,133,248,152
1330 DATA  133,247,38,251,38,252
     ,38,253,38,254
1340 DATA  38,247,38,248,202,208
     ,224,165,247,208
1350 DATA  4,165,248,240,7,162,0
     ,232,246,250
1360 DATA  240,251,160,1,165,253
     ,208,6,165,254
1370 DATA  208,2,160,0,132,80,24
     ,144,11,166
1380 DATA  87,202,134,87,224,255
     ,208,2,198,88
1390 DATA  165,91,133,78,165,92,
     133,79,24,165
```

(continua pagina seguente)

```
1400 DATA  89,101,251,133,89,165
     ,90,101,252,133
1410 DATA  90,165,91,101,253,133
     ,91,165,92,101
1420 DATA  254,133,92,32,141,193
     ,165,80,240,24
1430 DATA  166,78,232,134,78,224
     ,0,208,2,230
1440 DATA  79,234,165,78,197,91,
     208,231,165,79
1450 DATA  197,92,208,248,165,87
     ,197,93,208,175
1460 DATA  165,88,197,94,208,248
     ,96,165,88,240
1470 :
1480 REM  *****    BLOCCO 3    ***
     **
1490 :
1500 DATA  49552,49751,27606
1510 DATA  12,201,1,240,1,96,165
     ,87,201,64
1520 DATA  144,1,96,165,79,240,1
     ,96,165,78
1530 DATA  201,200,144,1,96,165,
     78,41,7,133
1540 DATA  247,165,78,74,74,41,2
     54,168,185,235
1550 DATA  193,56,229,247,133,24
     7,185,236,193,24
1560 DATA  101,88,133,248,165,87
     ,41,248,168,165
1570 DATA  87,41,7,76,36,194,193
     ,166,2,240
1580 DATA  5,17,247,145,247,96,7
     3,255,49,247
1590 DATA  145,247,96,128,64,32,
     16,8,4,2
1600 DATA  1,7,254,199,252,135,2
     51,71,250,7
1610 DATA  249,199,247,135,246,7
     1,245,7,244,199
1620 DATA  242,135,241,71,240,7,
     239,199,237,135
1630 DATA  236,71,235,7,234,199,
     232,135,231,71
1640 DATA  230,7,229,199,227,135
     ,226,71,225,7
1650 DATA  224,234,234,234,234,2
     34,234,234,74,170
1660 DATA  189,58,192,73,255,49,
     247,145,247,189
1670 DATA  58,192,166,2,61,62,19
     2,17,247,145
1680 DATA  247,96,169,71,141,0,3
     ,169,194,141
1690 DATA  1,3,96,134,254,169,32
     ,45,17,208
1700 DATA  240,9,169,14,133,251,
     169,6,32,207
1710 :
1720 REM  *****    BLOCCO 4    ***
     **
1730 :
1740 DATA  49752,49873,17109
1750 DATA  198,166,254,48,3,76,5
     8,164,76,116
1760 DATA  164,169,224,133,250,1
     69,0,133,249,162
1770 DATA  32,168,145,249,200,20
     8,251,230,250,202
1780 DATA  208,246,96,169,204,13
     3,250,160,0,132
1790 DATA  249,165,251,10,10,10,
     10,133,253,165
1800 DATA  252,41,15,24,101,253,
     162,4,145,249
1810 DATA  200,208,251,230,250,2
     02,208,246,96,165
1820 DATA  248,240,22,169,32,13,
     17,208,141,17
1830 DATA  208,169,56,141,24,208
     ,169,252,45,0
1840 DATA  221,141,0,221,96,169,
     223,45,17,208
1850 DATA  141,17,208,169,21,141
     ,24,208,169,3
1860 DATA  13,0,221,141,0,221,16
     5,251,141,134
1870 DATA  2,96
1880 :
1890 REM  *****    BLOCCO 5    ***
     **
1900 :
1910 DATA  49900,50099,28003
1920 DATA  169,0,133,97,133,99,2
     40,82,169,2
1930 DATA  133,97,133,99,169,0,1
     33,98,240,12
1940 DATA  169,1,133,97,169,0,13
     3,99,169,6
1950 DATA  133,98,165,93,133,254
     ,165,94,133,255
```

(continua pagina seguente)

```
1960 DATA  164,98,185,212,2,133,
     247,185,213,2
1970 DATA  133,248,169,0,133,81,
     32,131,195,165
1980 DATA  253,133,93,165,254,13
     3,94,165,95,133
1990 DATA  254,165,96,133,255,16
     9,1,133,81,32
2000 DATA  141,195,165,253,133,9
     5,165,254,133,96
2010 DATA  165,87,133,254,165,88
     ,133,255,173,212
2020 DATA  2,133,247,173,213,2,1
     33,248,169,0
2030 DATA  133,81,32,131,195,165
     ,253,133,87,165
2040 DATA  254,133,88,165,91,133
     ,254,165,92,133
2050 DATA  255,169,1,133,81,32,1
     41,195,165,253
2060 DATA  133,91,133,78,165,254
     ,133,92,133,79
2070 DATA  96,169,0,133,249,230,
     248,208,2,230
2080 DATA  249,165,255,133,80,16
     ,5,162,5,32
2090 DATA  0,192,169,0,133,253,1
     69,25,133,82
2100 DATA  169,0,133,252,133,251
     ,133,250,56,165
2110 DATA  250,229,247,168,165,2
     51,229,248,170,165
2120 :
2130 REM   *****    BLOCCO 6     ***
     **
2140 :
2150 DATA  50100,50189,12978
2160 DATA  252,229,249,144,6,133
     ,252,134,251,132
2170 DATA  250,38,253,38,254,38,
     255,38,250,38
2180 DATA  251,38,252,198,82,208
     ,217,165,80,16
2190 DATA  7,162,4,165,254,32,0,
     192,198,99
2200 DATA  16,29,165,81,208,4,16
     9,160,208,2
2210 DATA  169,100,24,101,253,13
     3,253,165,254,105
2220 DATA  0,133,254,80,6,198,25
     4,169,255,133
2230 DATA  253,96,13,198,32,198,
     57,198,66,198
2240 DATA  99,194,170,198,202,19
     8,234,198,42,199
2250 :
2260 REM   *****    BLOCCO 7     ***
     **
2270 :
2280 DATA  50215,50414,23966
2290 DATA  162,4,189,223,2,157,1
     68,2,202,16
2300 DATA  247,169,228,133,25,16
     9,2,133,26,76
2310 DATA  184,196,160,2,169,168
     ,32,162,187,32
2320 DATA  107,226,160,2,169,183
     ,32,40,186,32
2330 DATA  170,177,170,152,24,10
     9,190,2,141,203
2340 DATA  2,138,109,191,2,141,2
     04,2,160,2
2350 DATA  169,168,32,162,187,32
     ,100,226,160,2
2360 DATA  169,178,32,40,186,32,
     170,177,170,152
2370 DATA  24,109,188,2,141,201,
     2,138,109,189
2380 DATA  2,141,202,2,96,169,0,
     141,168,2
2390 DATA  141,169,2,141,170,2,1
     41,171,2,141
2400 DATA  172,2,169,226,133,26,
     169,229,133,25
2410 DATA  169,126,141,173,2,169
     ,32,141,174,2
2420 DATA  169,217,141,175,2,169
     ,123,141,176,2
2430 DATA  169,197,141,177,2,32,
     21,192,32,244
2440 DATA  194,164,93,165,94,32,
     145,179,162,178
2450 DATA  160,2,32,212,187,165,
     87,141,188,2
2460 DATA  165,88,141,189,2,164,
     95,165,96,32
2470 DATA  145,179,162,183,160,2
     ,32,212,187,165
2480 DATA  91,141,190,2,165,92,1
     41,191,2,169
2490 :
2500 REM   *****    BLOCCO 8     ***
```

(continua pagina seguente)

```
    **
2510 :
2520 DATA  50415,50614,23197
2530 DATA  0,133,181,32,61,196,1
     73,201,2,141
2540 DATA  195,2,173,202,2,141,1
     96,2,173,203
2550 DATA  2,141,199,2,173,204,2
     ,141,200,2
2560 DATA  160,2,169,168,32,162,
     187,160,2,169
2570 DATA  173,32,103,184,160,2,
     162,168,32,212
2580 DATA  187,165,25,164,26,32,
     91,188,201,255
2590 DATA  240,14,169,1,133,181,
     160,4,177,25
2600 DATA  153,168,2,136,16,248,
     32,61,196,162
2610 DATA  9,189,195,2,149,87,20
     2,16,246,165
2620 DATA  91,133,78,165,92,133,
     79,32,94,197
2630 DATA  32,72,192,32,109,197,
     165,181,240,152
2640 DATA  96,169,254,45,14,220,
     141,14,220,169
2650 DATA  253,37,1,133,1,96,169
     ,2,5,1
2660 DATA  133,1,169,1,13,14,220
     ,141,14,220
2670 DATA  96,162,36,138,160,0,2
     09,122,240,6
2680 DATA  32,138,173,162,0,96,3
     2,115,0,96
2690 DATA  32,124,197,224,36,240
     ,9,32,170,177
2700 DATA  140,208,2,141,209,2,3
     2,253,174,32
2710 DATA  124,197,224,36,240,9,
     32,170,177,140
2720 DATA  210,2,141,211,2,32,25
     3,174,32,124
2730 :
2740 REM  *****   BLOCCO 9   ***
     **
2750 :
2760 DATA  50615,50814,24444
2770 DATA  197,224,36,240,15,32,
     170,177,170,16
2780 DATA  3,76,72,178,140,212,2
     ,141,213,2
2790 DATA  96,32,253,174,32,124,
     197,224,36,240
2800 DATA  9,32,170,177,140,214,
     2,141,215,2
2810 DATA  32,253,174,32,124,197
     ,224,36,240,9
2820 DATA  32,170,177,140,216,2,
     141,217,2,96
2830 DATA  32,253,174,32,124,197
     ,224,36,240,15
2840 DATA  32,170,177,170,16,3,7
     6,72,178,140
2850 DATA  218,2,141,219,2,96,32
     ,143,197,32
2860 DATA  31,192,32,236,194,32,
     94,197,32,141
2870 DATA  193,32,109,197,96,32,
     143,197,32,204
2880 DATA  197,32,243,197,32,21,
     192,32,0,195
2890 DATA  32,94,197,32,72,192,3
     2,109,197,96
2900 DATA  32,143,197,32,126,198
     ,76,134,196,32
2910 DATA  143,197,32,126,198,32
     ,253,174,32,124
2920 DATA  197,224,36,240,7,162,
     223,160,2,32
2930 DATA  212,187,32,253,174,32
     ,124,197,224,36
2940 DATA  240,7,162,228,160,2,3
     2,212,187,32
2950 DATA  253,174,32,124,197,22
     4,36,240,7,162
2960 DATA  173,160,2,32,212,187,
     76,39,196,32
2970 :
2980 REM  *****   BLOCCO 10   ***
     **
2990 :
3000 DATA  50815,51014,27796
3010 DATA  204,197,173,215,2,16,
     3,76,72,178
3020 DATA  173,217,2,48,248,96,2
     34,234,32,158
3030 DATA  183,224,16,48,3,76,72
     ,178,134,252
3040 DATA  32,253,174,32,158,183
     ,224,16,16,241
3050 DATA  134,251,96,32,145,198
```

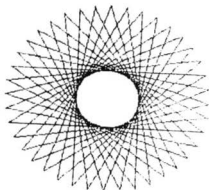
(continua pagina seguente)

```
 ,32,121,194,169
3060 DATA  239,45,22,208,141,22,
 208,169,170,141
3070 DATA  207,193,169,189,141,2
 08,193,169,227,141
3080 DATA  209,193,76,161,194,32
 ,145,198,165,252
3090 DATA  141,33,208,169,216,13
 3,250,160,0,132
3100 DATA  249,165,251,32,144,19
 4,169,239,45,22
3110 DATA  208,141,22,208,76,183
 ,194,32,145,198
3120 DATA  165,252,141,33,208,16
 5,251,133,253,32
3130 DATA  253,174,32,145,198,16
 9,216,133,250,160
3140 DATA  0,132,249,165,251,32,
 144,194,165,253
3150 DATA  133,251,32,121,194,16
 9,16,13,22,208
3160 DATA  141,22,208,169,76,141
 ,207,193,169,36
3170 DATA  141,208,193,169,194,1
 41,209,193,76,161
3180 DATA  194,32,158,183,224,4,
 48,3,76,72
3190 DATA  178,134,2,96,234,169,
 67,141,8,3
3200 DATA  169,199,141,9,3,96,32
 ,115,0,201
3210 :
3220 REM  *****  BLOCCO 11  ***
 **
3230 :
3240 DATA  51015,51163,17772
3250 DATA  95,240,6,32,121,0,76,
 231,167,169
3260 DATA  0,133,249,165,122,133
 ,252,165,123,133
3270 DATA  253,160,0,240,20,165,
 252,133,122,165
3280 DATA  253,133,123,230,249,1
 69,9,197,249,240
3290 DATA  218,200,202,208,252,1
 85,172,199,170,32
3300 DATA  115,0,133,254,200,185
 ,172,199,197,254
3310 DATA  208,219,202,208,240,6
 ,249,166,249,189
3320 DATA  252,195,141,63,3,232,
 189,252,195,141
3330 DATA  64,3,169,32,141,62,3,
 169,96,141
3340 DATA  65,3,32,115,0,32,62,3
 ,76,174
3350 DATA  167,4,80,76,79,84,4,6
 8,82,65
3360 DATA  87,6,67,73,82,67,76,6
 9,3,65
3370 DATA  82,67,5,67,76,69,65,8
 2,4,71
3380 DATA  82,65,70,4,84,69,88,8
 4,5,77
3390 DATA  71,82,65,70,4,67,79,7
 6,176
```

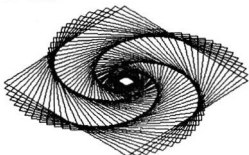
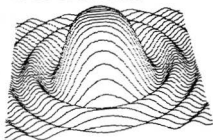
Esempi d'applicazione

```
100 REM  ****  RAGGIERA  ****
110 :
120 ←CLEAR:←GRAF11,13:A=1
130 ←COLOR  A
140 FOR I=0 TO 2*π STEP π/20
150 X2=COS(I)*110
160 Y2=SIN(I)*100
170 ←DRAW0,0,0,X2,Y2,0
180 NEXT
190 A=¬(A=0):GOTO 130
```

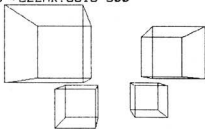
```
100 REM  ****  POLIGONI ****
110 :
120 ←CLEAR:←GRAF5,1
130 FOR L=3 TO 12
140 ←COLOR 1:←ARC0,0,0,110,100
 ,π/L,π/L+π*2,2*π/L
150 FOR K=0 TO 1000:NEXT: REM
 ** PAUSA **
160 ←COLOR 0:←ARC$,$,$,$,$,$,$
 ,$
170 NEXT
180 :
190 REM  ****  STELLE  ****
200 :
210 ←CLEAR:←COLOR 1
220 FOR L=.5*π TO 2.5*π STEP π/
 4
230 ←ARC0,0,0,110,100,L,L+4*π,4
 *π/5
240 FOR K=0 TO 1000:NEXT: REM
 ** PAUSA **
250 NEXT
260 STOP
```

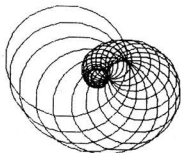
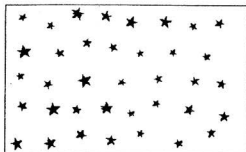
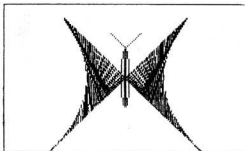
```
100 REM  **** COSINUSOIDI 3D **
    **
110 :
120 ←GRAF6,1:←CLEAR:←COLOR 1
130 FOR I=-3*π TO 3*π STEP 3*π/
    160
140 X=I*160/π/3:Y=COS(I)*16-84
150 FOR Z=0 TO 500 STEP 100
160 ←PLOTX,Y+Z/2.5,Z
170 NEXT
180 NEXT
190 FOR I=-3*π TO 3*π STEP π/2
200 X=I*160/π/3:Y=COS(I)*16-84
210 ←DRAWX,Y,0,X,Y+200,500
220 NEXT
230 STOP
```

```
100 REM  ****  CBM  ****
110 :
120 ←CLEAR:←MGRAF11,1,13,5
130 FOR Z=180 TO 0 STEP -3:←COL
    OR 1+Z/61
140 ←ARC-120,0,Z,50,100,π/4,2*π
    -π/4,π/2
150 :
160 ←ARC-45,36,Z,85,36,-π/2,π/2
    ,π/3
170 ←ARC-45,-36,Z,85,36,-π/2,π/
    2,π/3
180 ←DRAW-45,70,Z,-45,-70,Z
190 :
200 ←DRAW60,70,Z,60,-70,Z
210 ←DRAW150,70,Z,150,-70,Z
220 ←DRAW60,70,Z,105,25,Z
230 ←DRAW150,70,Z,105,25,Z
300 NEXT
500 STOP
```

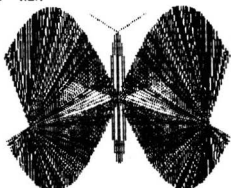
```
100 REM   ****   STELLA MULTICOL
    OR  ****
110 :
120 ←CLEAR:←MGRAF0,7,6,8
130 FOR AI=π/2 TO 2.5*π-.001 ST
    EP π/2
140 READ I,K,C
150 FOR J=I TO K STEP -1
160 ←COLOR C
170 ←ARC0,0,0,J*1.2,J,AI,AI+4*π
    ,4*π/5
180 NEXT:NEXT
190 DATA  100,65,1,75,65,2,85,6
    5,3,75,65,2
200 POKE 198,0:WAIT 198,1: REM
     ** ATTENDE LA PRESSIONE DI
     UN TASTO
```

```
100 REM   *** SPIRALE DI QUADRIL
    ATERI ***
110 :
120 ←CLEAR:←GRAF6,3:←COLOR 1
130 FOR I=0 TO 2*π STEP π/40
140 ←ARC0,0,I*I*80,160,100,I,I+
    2*π,π/2
150 NEXT
160 STOP
```

```
100 REM   **** FUNZIONE 3D ****
120 A=-3.5:B=-A:H=-99:D=10:S=40
    :F=92:K=5
130 DEF FNZ(W)=SIN(W)/W
140 DIM A(320):FOR I=0 TO 320:A
    (I)=-99999:NEXT
150 ←CLEAR:←COLOR 1:←GRAF11,5
160 FOR Z=A TO B STEP (B-A)/S
170 Z1=Z*Z+.0001:Z2=INT((Z-A)*D
    )
180 Z3=256/(Z2+256):X1=-160:H=H
    +K
190 FOR X=A TO B STEP (B-A)/319
200 Y=FNZ(X*X+Z1):X2=X1*Z3+160
210 Y1=(Y*F+H)*Z3
220 IF Y1<A(X2) THEN 250
230 A(X2)=Y1
240 ←PLOTX1,Y1/Z3,Z2
250 X1=X1+1:NEXT:NEXT
```

```
100 REM   *** IN MOVIMENTO NEL T
    UNNEL ***
110 :
120 ←MGRAF0,0,0,13:←CLEAR:POKE
    53280,11
130 L=70:Z=1200:Y=99
140 FOR I=0 TO 3
150 FOR C=1 TO 3:←COLOR C:Z=Z-
    100
160 FOR X=-155 TO 155 STEP 310
170 ←DRAWX,Y,Z,X,Y-200,Z
180 ←DRAW$,$,Z+L,$,$,Z+L
190 ←DRAW$,Y-200,$,$,$,Z
200 ←DRAW0,$,$,0,$,$
210 NEXT
220 FOR X=-155 TO 85 STEP 80
230 ←DRAWX,Y,Z,X+L,Y,Z
240 ←DRAW$,$,Z+L,$,$,Z+L
250 ←DRAWX+L,$,Z,$,$,$
260 ←DRAWX,$,$,X,$,$
270 NEXTX,C,I
280 :
290 A=100:B=0:S=-4: REM  ** VIA
     ! **
300 FOR I=1 TO 3
310 FOR T=A TO B STEP S
320 FOR C=1 TO 3
330 ←MGRAF0,13,0,0
340 FOR J=0 TO T:NEXT
350 ←MGRAF0,0,13,0
360 FOR J=0 TO T:NEXT
370 ←MGRAF0,0,0,13
380 FOR J=0 TO T:NEXT
390 NEXTC,T
400 IF I=1 THEN A=0:B=1:S=.1
410 IF I=2 THEN A=0:B=100:S=10
420 NEXTI
430 GOTO 290
```

```
100 REM   ***     CUBO NELLO SPAZ
    IO      ***
110 :
120 REM  *** L=LATO  X=POSIZ. O
    RIZZONTALE  Y=POSIZ. VERTIC
    ALE  Z=DISTANZA ***
130 REM  *** X,Y,Z SI RIFERISCO
    NO ALLO SPIGOLO IN ALTO A S
    INISTRA ***
140 REM  *** PIU' "VICINO" ALLO
     SCHERMO ***
150 :
160 A$="[DOWN][5 RIGHT]":←TEXT6
```

```
    ,3
170 PRINT"[CLEAR]"A$;A$"[RVS] C
    UBO NELLO SPAZIO "
180 PRINTA$"[3 DOWN]PER SPOSTAR
    E IL CUBO SU, GIU'"
190 PRINTA$"A DESTRA E A SINIST
    RA"
200 PRINTA$"USARE I TASTI DEL C
    URSORE"
210 PRINTA$"[DOWN]PER ALLONTANA
    RLO E AVVICINARLO"
220 PRINTA$"USARE I TASTI - E +
    "
230 PRINTA$;A$"[DOWN][RVS] PREM
    ERE UN TASTO "
240 GET A$:IF A$="" THEN 240
250 :
260 ←CLEAR:←GRAF0,1:←COLOR 1
270 POKE 650,128
280 L=200:X=170:Y=-75:Z=500
290 :
300 FOR Y1=Y TO Y-L STEP -L
310 ←DRAWX,Y1,Z,X+L,Y1,Z
320 ←DRAW$,$,Z+L,$,$,Z+L
330 ←DRAWX+L,$,Z,$,$,$
340 ←DRAWX,$,Z,X,$,$
350 NEXT
360 FOR Z1=Z TO Z+L STEP L
370 ←DRAWX,Y,Z1,X,Y-L,Z1
380 ←DRAWX+L,$,$,X+L,$,$
390 NEXT
400 :
410 POKE 198,0:WAIT 198,1:GET A
    $
420 IF A$="-" THEN Z=Z+100
430 IF A$="+" AND Z>99 THEN Z=Z
    -100
440 IF A$="[UP]" THEN Y=Y+50
450 IF A$="[DOWN]" THEN Y=Y-50
460 IF A$="[RIGHT]" THEN X=X+50
470 IF A$="[LEFT]" THEN X=X-50
480 ←CLEAR:GOTO 300
```

```
100 REM    ****  PAESAGGIO  ****
110 :
120 POKE 53280,11:←CLEAR:←MGRAF
    13,9,7,14
130 ←COLOR 3:FOR Y=0 TO 100:←D
    RAW-160,Y,0,160,Y,0:NEXT:RE
    M  ** CIELO
140 :
150 ←COLOR 2:FOR R=13 TO 2 STE
    P -1:←CIRCLE-100,85,0,R*1.2
    ,R:NEXT:REM  ** SOLE
160 FOR I=π/2 TO 2.4*π STEP π/8
170 Y=SIN(I)*32+85:X=COS(I)*43-
    100
180 ←DRAW-100,85,0,X,Y,0
190 NEXT:REM  ** RAGGI
200 :
210 ←COLOR 1:A=-150:B=A
220 M=2*INT(15+15*RND(1))
230 FOR Y=M TO 0 STEP -2:A=A-2*
    RND(1)-1:B=B+2*RND(1)+1
240 ←DRAWA,Y,0,B,Y,0:NEXT
250 IF B<180 THEN A=B+RND(1)*10
    :B=A:GOTO 220:REM  ** MONTI
260 :
270 ←COLOR 1:FOR X=-150 TO 90
    STEP 2:←DRAWX,-100,32000,X,
    -100,0:NEXT:REM  ** STRADA
280 :
290 FOR Z=800 TO 200 STEP -600
300 ←COLOR 1:FOR X=130 TO 160
    STEP 2:←DRAWX,-100,Z,X,100,
    Z:NEXT
310 S=-4:IF Z=200 THEN S=-1.7
320 ←COLOR  0:FOR R=58 TO 1 STE
    P S:←CIRCLE145,112,Z,R*1.7,
    R:NEXT
330 NEXT:REM  ** ALBERI
340 POKE 198,0:WAIT 198,1:REM
    ** ATTENDE TASTO PREMUTO
```

```
100 REM  ****  CHIOCCIOLA  ****
110 :
120 ←GRAF11,7:←COLOR 1:←CLEAR
130 FOR I=2.5*π TO 0 STEP -π/10
140 X=COS(I)*(80-I*I)-40
150 Y=SIN(I)*(70-I*I)-24
160 ←CIRCLEX,Y,I*I*25,80,74
170 NEXT
180 STOP
```

```
100 REM  ***  CIELO STELLATO  *
    **
110 :
120 ←CLEAR:←COLOR 1:←GRAF0,1
130 FOR Y1=-80 TO 80 STEP 40
140 FOR X=-180 TO 100
150 X=RND(2)*20+30+X:Y=Y1+RND(0
    )*20-10
160 AI=RND(1)*2:W=RND(1)*5+5
170 FOR I=1 TO W
180 ←ARCX,Y,0,I,I,AI,AI+4*π,4*π
    /5
190 NEXTI,X,Y1
200 STOP
```

```
100 REM  *** FARFALLA MULTICOLO
    R ***
110 :
120 ←MGRAF6,1,10,0:←CLEAR
130 ←COLOR 1
140 FOR I=2 TO 5 STEP 3
150 ←CIRCLE0,0,0,I,40
160 NEXT
170 ←DRAW0,40,0,-20,60,0
180 ←DRAW0,40,0,20,60,0
190 A=100
200 FOR I=0 TO A*.8 STEP 2
210 ←COLOR  RND(0)*3+1
220 ←DRAW I-A,I-A,0,-I,I,0
230 ←DRAW A-I,I-A,0,I,I,0
240 NEXT
250 FOR I=0 TO 4000:NEXT
260  RUN
```

```
100 REM  *** FARFALLA MULTICOLO
    R ***
110 :
120 ←MGRAF6,1,10,0:←CLEAR
130 ←COLOR 1
140 FOR I=2 TO 8 STEP 3
150 ←CIRCLE0,0,0,I,70
160 NEXT
170 ←DRAW-30,90,0,0,70,0
180 ←DRAW30,90,0,0,70,0
190 :
200 FOR I=-.85*π TO 0 STEP π/80
210 X=I*45:Y=SIN(I)*99
220 ←COLOR  RND(0)*3+1
230 FOR K=-1 TO 1 STEP 2
240 ←DRAW0,0,0,X*K,Y*K,0
250 ←DRAW95*K,-30,0,-X*K,Y*K,0
260 ←DRAW0,0,0,X*K,-Y*K,0
270 ←DRAW95*K,-30,0,-X*K,-Y*K,0
280 NEXTK,I
```

```
290 FOR I=0 TO 5000:NEXT
300  RUN
```

```
100 REM  **** RAGNATELA ****
110 :
120 ←GRAF11,13:←CLEAR:←COLOR 1
130 L=100:N=16
140 FOR J=0 TO N-1
150 FOR I=0 TO J-1
160 A=2/N*π
170 X1=L*COS(A*J):Y1=L*SIN(A*J)
180 X2=L*COS(A*I):Y2=L*SIN(A*I)
190 ←DRAWX1,Y1,0,X2,Y2,0
200 NEXTI,J
210 STOP
```

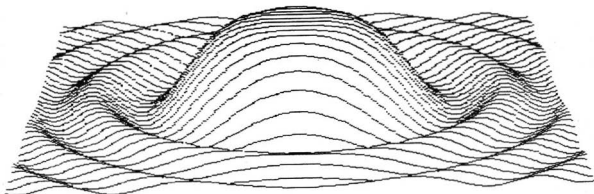
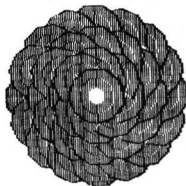
```
100 REM  **** SPIRALI ****
110 :
120 ←GRAF11,13:←CLEAR:←COLOR 1
130 FOR I=-π TO π STEP π/60
140 X1=100*COS(I/2):Y1=100*SIN(
    I*2)
150 X2=100*COS(I*2):Y2=100*SIN(
    I/2)
160 ←DRAWX1,Y1,0,X2,Y2,0
170 NEXT
180 STOP
```

```
100 REM  ***  DEMO TERZA DIMENS
    IONE  ***
110 :
120 ←CLEAR:←GRAF11,0:←COLOR 1
130 FOR Z=0 TO 1800 STEP 300
140 ←ARC0,0,Z,200,100,-.25*π,1.
    25*π,π/2
150 NEXT
160 FOR X=-140 TO 140
170 ←DRAWX,-71,0,X,-71,1800
180 NEXT
190 STOP
```

```
100 REM  ** QUADRATO CHE SI ALL
    ONTANA **
110 :
120 REM  *** L=LATO  X=POSIZ. O
    RIZZONTALE  Y=POSIZ. VERTIC
    ALE  Z=DISTANZA ***
130 :
140 ←GRAF11,3:←COLOR 1
150 L=100:Y=-100
160 X=RND(0)*300-200
170 :
180 FOR Z=0 TO 2000 STEP 100
190 ←CLEAR
200 ←DRAWX,Y,Z,X+L,Y,Z
210 ←DRAW$,$,Z+L,$,$,Z+L
220 ←DRAWX+L,$,Z,$,$,$
230 ←DRAWX,$,Z,X,$,$
240 NEXT
250 GOTO 160
```

```
100 REM  *** GRAFICI DI FUNZION
    E ***
110 :
120 POKE 53280,6
130 PRINT"[CLEAR][10 RIGHT][RVS
    ]GRAFICI DI FUNZIONE[RVOFF]
    "
140 PRINTSPC(217)"PREMI":PRINTS
    PC(86)"[RVS]1[RVOFF] PER SC
    EGLIERE L'INTERVALLO"
150 PRINTSPC(46)"[RVS]2[RVOFF]
    PER CAMBIARE FUNZIONE"
160 GET A$:ON VAL(A$)GOTO 210,1
    80
170 GOTO 160
180 PRINT"[CLEAR]MODIFICA LA SE
    GUENTE LINEA"
190 PRINT"[DOWN]POI DAI IL RUN[
    3 DOWN]"
200 LIST 240
210 INPUT "[CLEAR][3 DOWN][RVS]
     ESTREMI DELL' INTERVALLO[R
    VOFF] ";A,B
220 M=99999:L=-M
230 S=(B-A)/319
240 DEF FNY(X)=SIN(X)
250 FOR X=A TO B STEP S:Y=FNY(X
    )
260 IF Y>L THEN L=Y
270 IF Y<M THEN M=Y
280 NEXT
290 PRINT"[2 DOWN][RVS] MASSIMO
```

```
    DELL' INTERVALLO[RVOFF]";L
300 PRINT"[2 DOWN][RVS] MINIMO
    DELL' INTERVALLO [RVOFF]";M
310 INPUT "[2 DOWN][RVS] SCALA
    DI Y              [RVOFF] "
    ;P
320 IF P<>0 THEN P1=S/P
330 T=(L-M)/S/199:IF P=0 THEN P
    =T:P1=S*P
340 ←CLEAR:←GRAF6,3:←COLOR 1
350 IF M>0 THEN E=-M/S-100
360 IF L<0 THEN E=-L/S+100
370 IF M*L<=.01 THEN GOSUB 460
380 IF A*B<=0 THEN GOSUB 490
390 I=-160:FOR X=A TO B STEP S:
    Y=E+FNY(X)/P1:IF Y>100 OR Y
    <-100 THEN 410
400 ←PLOTI,Y,0
410 I=I+1:NEXT
420 GET A$:IF A$=""GOTO 420
430 ←TEXT6,14
440 GOTO 120
450 :
460 E=100-L/S/T:IF E>100 THEN E
    =100
470 IF E<-100 THEN E=-100
480 ←DRAW-160,E,0,160,E,0:RETUR
    N
490 X=-A/S-160:←DRAWX,-100,0,X,
    100,0:RETURN
```

```
100 REM  ****  FIORE  ****
110 :
120 ←CLEAR:←MGRAF7,4,2,0
140 FOR N=4 TO 1 STEP -1
150 FOR I=0 TO 2*π-.01 STEP π/2
    /N
160 X=COS(I)*23*(N-1)
170 Y=SIN(I)*23*(N-1)
180 FOR A=10 TO 30:←COLOR A/10
190 ←ARCX,Y,0,A,A,I-π/1.5,I+π/1
    .5,2*π/10
200 NEXTA,I,N
400 WAIT 198,1
```

Neutralizzazione del tasto Restore

Come già spiegato nel paragrafo riguardante le caratteristiche generali delle routine, il tasto Restore provoca catastrofici effetti se viene premuto mentre è disabilitato il Sistema Operativo (quindi mentre è in esecuzione un'istruzione di tracciamento come DRAW, ARC, e così via). Si ottiene infatti il blocco del computer, rimediabile solo con lo spegnimento.

Quest'inconveniente può essere fonte di grosse arrabbiature, soprattutto per i più distratti, perciò vi farà piacere poterlo eliminare.

Il tasto maledetto

Vediamo di capire come funziona il temibile tasto.

Quando premiamo Restore provochiamo un NMI, vale a dire un interrupt non mascherabile, che per il nostro microprocessore rappresenta "un ordine tassativo" (gli esperti di elettronica perdonino la terminologia poco ortodossa) che non può rifiutarsi di "ascoltare".

Perciò interrompe qualunque cosa stia facendo e si precipita ad eseguire la routine di NMI del Sistema Operativo, il cui indirizzo è contenuto nelle locazioni FFFA e FFFB (valori esadecimali corrispondenti ai decimali 65530 e 65531).

Questa prevede il test del tasto Run/Stop. Se, al momento della verifica, risulta premuto, avremo come effetto alcune riinizializzazioni: delle periferiche, dei vettori principali e dello stack; subito dopo comparirà la solita scritta READY nella parte alta dello schermo.

Se invece il tasto Run/Stop non è premuto, il microprocessore tornerà ad occuparsi di ciò che stava facendo prima dell'interruzione, con l'unica conseguenza di avere "perso" alcuni microsecondi.

Con le routine di Toma, però, durante il tracciamento viene disabilitata la ROM che contiene il Sistema Operativo e quindi dalla locazione E000 (decimale 57344) alla FFFF (decimale 65535) il microprocessore trova solo memoria RAM.

Quando, pertanto, in seguito alla pressione del tasto Restore, cerca l'indirizzo della routine di NMI... non trova ciò che cerca e va in tilt.

Disabilitare il tasto Run/Stop è inutile (poichè sarebbe come "chiudere la stalla dopo che i buoi sono scappati").

La soluzione comunque è molto semplice: basta creare una routine di NMI "alternativa" in una zona sicura e mettere nelle locazioni (della RAM) FFFA e FFFB l'indirizzo di tale routine.

Così anche nel caso in cui il Sistema Operativo sia momentaneamente "assente", il microprocessore saprà dove andare e cosa fare.

La nuova routine di NMI creata è il massimo della semplicità. E' formata da una sola istruzione in linguaggio macchina: RTI che, per i profani, significa "ritorno da interrupt".

Ciò significa che ogni richiesta di NMI attiva la solita procedura ma non provoca nessuna conseguenza, anche se il tasto STOP è stato premuto, perchè al microprocessore viene subito imposto di tornare alla sua attività ante-interrupt.

L'unico effetto è la solita perdita di microsecondi per effettuare i vari salti.

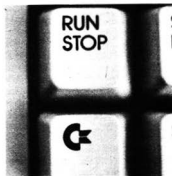
La "procedura alternativa" qui descritta viene seguita SOLAMENTE quando la ROM del Sistema Operativo è disabilitata. Non si ha perciò nessuna modifica nella funzione del tasto Restore al di fuori del caso citato.

Le modifiche da apportare

Il lavoro da compiere per una modifica di tale importanza è decisamente leggero: basta infatti inserire le linee 10000-10200 (vedi programma: "Disabilita Restore") in fondo al listato delle routine grafiche che avete già digitato. La numerazione, infatti, consente l'accodamento del nuovo blocco di righe Basic.

Le tre POKE modificano la routine di disabilitazione del Sistema Operativo quindi vanno effettuate DOPO avere letto e memorizzato le routine grafiche, altrimenti non si avrà alcun effetto.

Per tale motivo nel listato devono assolutamente essere poste in modo da essere eseguite dopo la lettura dei DATA delle già citate routine.

Aggiungendo le poche righe di programma Basic al gruppo di righe del listato "Routine Grafiche" si eviterà, ad ogni buon conto, ogni perplessità o errore.

```
9940 REM ************************
9950 REM *                      *
9960 REM * DISABILITA RESTORE *
9970 REM *                      *
9980 REM ************************

9990 :
```

```
10000 FOR I-49874 TO 49887:READ A
      :POKE I,A:NEXT
10010 DATA  133,1,169,223,141,250
      ,255,169,194,141,251,255,96
      ,64
10020 POKE 50538,76:POKE 50539,21
      0:POKE 50540,194
```

Comandi per il salvataggio/caricamento della pagina grafica

Vediamo ora quattro istruzioni che si aggiungono alle nove prima viste e che saranno senz'altro utili a molti "artisti". Infatti salvare su nastro (o su disco) i propri "capolavori", disegnati con le routine grafiche, risulta molto difficile per tutti coloro che non conoscono approfonditamente il Sistema Operativo e la configurazione di memoria del nostro C/64.

Effettivamente, dal momento che la pagina grafica è situata in una posizione particolarmente inaccessibile, l'operazione presenta alcune difficoltà da superare.

Salvare direttamente tramite i normali comandi Basic (OPEN oppure SAVE usato nel modo atto a salvare il contenuto di una zona di memoria da voi scelta) non è possibile. Infatti la pagina grafica condivide la propria "casa" con il Sistema Operativo e quando cercate di leggere (con PEEK oppure, caso nostro, tramite SAVE) il contenuto delle locazioni in questione coincide con il Sistema Operativo stesso. Quindi il valore "peekato" non ha niente a che vedere con il contenuto dalla pagina grafica che si trova "sotto".

Stranamente (ma non troppo) l'operazione inversa, vale a dire POKE, non presenta inconvenienti di sorta perchè la POKE è sempre diretta alla R.A.M. (quindi alla pagina grafica) anche se "sopra" di essa è presente una ROM (il Sistema Operativo).

Ecco dunque la necessità di ricorrere ad una routine in linguaggio macchina che "tolga di mezzo" temporaneamente il Sistema Operativo e renda "visibile", in fase di lettura, la pagina grafica.

Ma se il Sistema Operativo viene disabilitato non è più possibile utilizzarne le comode routine dedicate alla comunicazione con le periferiche: risolto un problema ecco dunque che se ne presenta subito un altro!

Le soluzioni sono due: o si riscrivono, con le opportune modifiche, le preziose routine di I/O (ci vengono i brividi solo a pensarlo) oppure si trasferisce il contenuto della pagina grafica in una zona più accessibile e si riabilita il Sistema Operativo. Esclusa, per motivi morali, la soluzione masochista, si è optato per la seconda soluzione.

Ma dove trasferire 8K (circa) di memoria senza rischiare di invadere lo spazio riservato ai programmi in Basic? Naturalmente (!) sotto la R.O.M dell'interprete Basic.

Questa è "mappata" dalla locazione 40960 alla locazione 49151 e non interviene nelle operazioni di I/O. Può quindi venire disabilitata senza importanti conseguenze. Questo trasferimento della pagina ha permesso, inoltre, di realizzare l'istruzione GRMERGE (vedi dopo) che sicuramente vi farà comodo.

Quattro nuovi comandi

Ricordiamo che anche questi comandi vanno preceduti dal solito carattere di freccia a sinistra.

GRSAVE
La sintassi dei parametri è identica a quella del SAVE "normale". Esempio:

GRSAVE
GRSAVE"nome"
GRSAVE"nome",n
(n=numero di periferica)

La pagina grafica viene salvata come un normale file programma (PRG).

Questa, come le altre istruzioni, funzionano sia con il registratore (n=1) che con il disk drive (n=8).

GRLOAD
Sintassi dei parametri identica al LOAD normale. Esempio:

GRLOAD
GRLOAD"nome"
GRLOAD"nome",n
GRLOAD"nome",n,1

L'indirizzo secondario non è necessario in quanto la pagina grafica caricata con questa istruzione viene SEMPRE messa negli ultimi 8K di memoria Ram.

E' ovvio che eventuali disegni presenti nella pagina al momento del GRLOAD vengono irrimediabilmente cancellati.

GRVERIFY
Sintassi dei parametri identica al VERIFY normale. Esempio:

GRVERIFY
GRVERIFY"nome"
GRVERIFY"nome",n

La verifica non viene effettuata comparando la registrazione con la pagina grafica "normale", ma con la sua immagine trasferita nelle locazioni 40960-48959. Immagine che rimane invariata fino al prossimo GRSAVE o GRLOAD.

GRMERGE
Sintassi dei parametri identica al LOAD. Esempio:

GRMERGE
GRMERGE"nome"
GRMERGE"nome",n
GRMERGE"nome",n,1

Stesse considerazioni di GRLOAD riguardo l'indirizzo secondario.

Questa istruzione consente di caricare dalla memoria di massa la pagina grafica e di sovrapporla a quella già presente nella memoria del calcolatore SENZA CANCELLARE quanto è già presente nella pagina grafica al momento del caricamento. Si ha cioè la possibilità di fondere le due pagine: quella residente e quella caricata.

Effetti collaterali

Se usate il registratore e premete il tasto Run/Stop quando compare la scritta "PRESS PLAY..." non succede nulla. Per fermare l'operazione dovete premere anche il tasto Restore.

Gli altri piccoli inconvenienti sagnalati nel numero 17 di Commodore Computer Club sono stati eliminati in questa nuova versione grazie all'aggiunta di due istruzioni: PHA e PLA, nella routine di riabilitazione del Basic.

Importante: nella frazione di secondo in cui è disabilitato il Sistema Operativo (con GRSAVE subito dopo che premete il tasto Return, con GRLOAD e GRMERGE alla fine del caricamento) non premete assolutamente il tasto Restore se non avete provveduto ad aggiungere alle routine grafiche la routine che lo disabi-

lita, perchè otterreste il blocco del computer.

Se avete provveduto ad aggiungere le poche righe di "Disabilita Restore" al corpo principale del programma, non avrete, però, nulla da temere.

Il listato

Per disporre delle quattro nuove istruzioni agite in questo modo:

1: Caricate le routine grafiche (alle quali avete "appeso" il programma "Disabilita Restore").

2: Digitate quindi il listato "Load/Save della pagina grafica", registrate e verificate.

3: Date il RUN.

4: Verificate il corretto funzionamento disegnando qualcosa in Hi-Res e, in seguito, registrando e caricando la pagina grafica servendosi dei nuovi comandi.

La locazione 51051 (riga 15350) indica il numero di nuovi comandi attivati, che passa da 9 a 13.

Al termine delle operazioni avrete un unico listato Basic che, una volta attivato, metterà a disposizione i 13 comandi grafici.

```
14870 REM ********************
14880 REM *                  *
14890 REM * COMANDI LOAD/SAVE *
14900 REM *  PAGINA  GRAFICA  *
14910 REM *                  *
14920 REM * INDIRIZZI:       *
14930 REM * GRSAVE=51418     *
14940 REM * GRMERGE=51468    *
14950 REM * GRLOAD=51472     *
14960 REM * GRVERIFY=51489   *
14970 REM *                  *
14980 REM ********************
14990 :
15000 B=0:FOR I=51380 TO 51561:RE
      AD A:B=B+A:POKE I,A:NEXT
15010 IF B<>24860 THEN PRINT"ERRO
      RE NEI DATA DELLE ROUTINES"
      :END
15020 B=0:FOR I=51164 TO 51183:RE
      AD A:B=B+A:POKE I,A:NEXT
15030 IF B<>1440 THEN PRINT"ERROR
      E NEI DATA DELLE PAROLE":EN
      D
15040 B=0:FOR I=50190 TO 50197:RE
      AD A:B=B+A:POKE I,A:NEXT
15050 IF B<>1082 THEN PRINT"ERROR
      E NEI DATA DEGLI INDIRIZZI"
      :END
15060 :
15070 REM      ****     ROUTINES
      ****
15080 DATA  169,0,133,249,133,251
      ,162,32,168,32
15090 DATA  94,197,32,88,201,177,
      249,36,251,145
15100 DATA  251,200,208,247,230,2
      50,230,252,202,208
15110 DATA  240,32,109,197,32,96,
      201,96,104,104
15120 DATA  169,224,133,250,169,1
      60,133,252,169,36
15130 DATA  141,197,200,32,180,20
      0,32,212,225,32
15140 DATA  88,201,162,64,160,191
      ,169,160,133,254
15150 DATA  169,0,133,253,169,253
      ,32,216,255,32
15160 DATA  96,201,144,3,76,249,2
      24,96,169,17
15170 DATA  208,2,169,36,141,197,
      200,169,224,133
15180 DATA  252,169,160,133,250,1
      69,0,240,2,169
15190 DATA  1,133,10,104,104,32,2
      12,225,32,88
15200 DATA  201,165,10,162,0,160,
      160,32,213,255
15210 DATA  32,96,201,176,205,165
      ,10,240,10,162
15220 DATA  28,32,183,255,41,16,2
      08,13,96,32
15230 DATA  180,200,32,183,255,41
      ,191,240,245,162
15240 DATA  29,76,55,164,120,169,
      254,37,1,133
15250 DATA  1,96,72,169,1,5,1,133
      ,1,88,104,96
15260 :
15270 REM      ****     PAROLE    ***
      *
15280 DATA  3,71,82,148,3,71,82,1
      47,3,71
15290 DATA  82,149,7,71,82,77,69,
      82,71,69
15300 :
15310 REM      ****     INDIRIZZI
      ****
15320 DATA  218,200,16,201,33,201
      ,12,201
15330 :
15340 REM   ** ATTIVA I NUOVI COMA
      NDI **
15350 POKE 51051,13
```

Comandi per scrivere nella pagina grafica

Con tre nuovi comandi, da aggiungere ai 13 precedenti, si completa il set d'istruzioni grafiche proposto.

Tramite le nuove istruzioni sarà possibile scrivere caratteri alfanumerici direttamente nella pagina grafica, sia in orizzontale che in verticale. Tutto ciò che vorremo (numeri, lettere, caratteri semigrafici) possono quindi integrare la leggibilità di qualsiasi disegno riprodotto.

I caratteri speciali (Home, Delete, Cursor, e così via) verranno visti come Cursor Right, faranno cioè tutti avanzare il nostro cursore ideale di uno spazio a destra.

CHAR
sint.: CHAR X,Y,T,⟨testo⟩
funz.: scrive la stringa o il numero specificati nella pagina grafica, partendo dalla colonna X della riga Y.

Se il messaggio non è numerico va messo tra virgolette come nell'istruzione PRINT.

Valori accettabili:

0⟨=X⟨=39
0⟨=Y⟨=24

Il terzo parametro (T) determina quale set di caratteri utilizzare:

T=0 upper case
T=1 reverse upper case
T=2 lower case
T=3 reverse lower case

Se un messaggio eccede il limite inferiore dello schermo, viene troncato automaticamente.

VCHAR
sint: VCHAR X,Y,T,⟨testo⟩
funz.: come CHAR ma il messaggio viene scritto verticalmente.

INV

sint.: INV N
funz.: determina il modo di disegno, normale se il valore di N è 0, ad inversione (in pratica viene effettuato un OR esclusivo dei punti) se tale valore è compreso tra 1 e 255.

In modo hi-res monocromatico, se viene settato COLOR 1, tutte le istruzioni di tracciamento eseguite dopo il comando INV1 invertono i punti da disegnare: se sono spenti li accendono, se sono accesi li spengono.

Con COLOR 0, invece, i punti vengono sempre spenti.

In modo multicolore vengono spenti solo i punti che presentano lo stesso colore dell'ultimo comando COLOR, mentre gli altri cambiano colore. La regola si ricava effettuando l'OR esclusivo tra le coppie di bit dei colori.

Problemi di visualizzazione possono presentarsi per linee orizzontali in multicolore a causa della larghezza doppia dei punti.

Per aggiungere i tre nuovi comandi procedete in questo modo:

1: Caricate le routine grafiche, che DEVONO ASSOLUTAMENTE comprendere anche i comandi per il caricamento e registrazione della pagina grafica (cfr. paragrafo precedente).
2: Digitate la routine di questo paragrafo ("Char/Inv"), registrate, verificate e date il RUN.

L'istruzione POKE 51051,16 di riga 20350 servirà ad indicare all'interprete che da quel momento sono 16 i nuovi comandi disponibili.

Anche in questo caso il listato viene "appeso" corpo principale del programma.

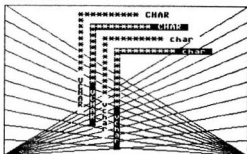
Potete cancellare la riga 15350 che, comunque, viene neutralizzata dalla nuova riga Basic 20350.

```
19780 REM ******************
19790 REM *                *
19800 REM * COMANDI   PER  *
19810 REM * SCRIVERE NELLA *
19820 REM * PAGINA GRAFICA *
19830 REM * ED EFFETTUARE  *
19840 REM * LA FUNZ. XOR   *
19850 REM *  DEI   PUNTI   *
19860 REM * DA TRACCIARE   *
19870 REM ******************
19880 REM * INDIRIZZI      *
19890 REM * CHAR=51568     *
19900 REM * VCHAR=51574    *
19910 REM * INV=51795      *
19920 REM ******************
19930 REM * INDISPENSABILE *
19940 REM * CARICARE PRIMA *
19950 REM * I COMANDI DI   *
19960 REM * SAVE/LOAD      *
19970 REM * PAGINA GRAFICA *
19980 REM ******************
19990 :
20000 B=0:FOR I=51568 TO 51826:RE
      AD A:B=B+A:POKE I,A:NEXT
20010 IF B<>34524 THEN PRINT"ERRO
      RE NEI DATA DELLE ROUTINE":
      END
20020 B=0:FOR I=51184 TO 51198:RE
      AD A:B=B+A:POKE I,A:NEXT
20030 IF B<>907 THEN PRINT"ERRORE
       NEI DATA DELLE PAROLE":END
20040 B=0:FOR I=50198 TO 50203:RE
      AD A:B=B+A:POKE I,A:NEXT
20050 IF B<>917 THEN PRINT"ERRORE
       NEI DATA DEGLI INDIRIZZI":
      END
20060 :
20070 REM   ****   ROUTINE   ****
20080 DATA  169,8,162,0,240,4,169
      ,64,162,1,133,181,134,182
20090 DATA  32,158,183,224,40,144
      ,3,76,72,178,138,162,0,32
```

```
20100 DATA  69,202,32,241,183,224
      ,25,176,240,134,249,169,24,
      56
20110 DATA  229,249,10,168,24,185
      ,235,193,101,247,41,248,133
      ,247
20120 DATA  185,236,193,101,248,1
      33,248,32,241,183,224,4,176
      ,207
20130 DATA  134,251,6,251,6,251,3
      2,253,174,32,158,173,36,13
20140 DATA  48,6,32,221,189,32,13
      5,180,32,166,182,170,240,10
      3
20150 DATA  133,252,160,0,132,253
      ,32,32,202,144,37,162,2,32
20160 DATA  69,202,169,208,24,101
      ,251,101,250,133,250,120,16
      9,251
20170 DATA  37,1,133,1,160,7,177,
      249,145,247,136,16,249,169
20180 DATA  4,5,1,133,1,88,198,25
      2,240,51,24,165,181,101
20190 DATA  247,133,247,165,182,1
      01,248,133,248,176,10,201,2
      55,208
20200 DATA  191,165,247,201,64,14
      4,185,96,164,253,177,34,200
      ,132
20210 DATA  253,170,48,16,201,32,
      144,11,201,96,144,4,41,223
20220 DATA  208,2,41,63,56,96,41,
      127,201,127,208,2,169,94
20230 DATA  201,32,96,160,0,148,2
      48,10,10,54,248,10,54,248
20240 DATA  149,247,96,32,158,183
      ,138,240,12,169,81,141,215,
      193
20250 DATA  141,45,194,141,55,194
      ,96,169,17,141,215
20260 DATA  193,141,55,194,169,14
      5,141,45,194,96
20270 :
20280 REM    ****    PAROLE    ****
20290 DATA  4,67,72,65,82,5,86,67
      ,72,65,82,3,73,78,86
20300 :
20310 REM    ****    INDIRIZZI    **
      **
20320 DATA  112,201,118,201,83,20
      2
20330 :
20340 REM  ** NUMERO DI COMANDI A
      TTIVATI **
20350 POKE 51051,16
```

```
100 REM   *** ARTE ASTRATTA ***
110 :
120 ←COL OR 1:←CLEAR:←GRAF8,0
130 ←INV1
200 X=-160:Y=-100
210 DEF FNR(K)=RND(0)*30*K+10
220 FOR I=0 TO 30
230 X1=X+FNR(5):X2=X1+FNR(5)
240 FOR Y1=Y+FNR(3) TO Y1+FNR(3
    )
250 ←DRAWX1,Y1,0,X2,Y1,0
260 NEXTY1,I
270 FOR I=0 TO 5000:NEXT
280  RUN
```

```
100 REM   ***   DEMO INV   ***
110 :
120 ←COL OR 1:←CLEAR:←GRAF3,11
130 ←CHAR11,10,0,"QUESTO E' L'E
    FFETTO"
140 ←CHAR12,14,1," DEL COMANDO
    INV "
150 ←INV1
160 FOR K=0 TO 2
170 FOR I=-90 TO 90
180 ←DRAWI,I,0,I,-I,0
190 NEXTI,K
200 ←INV0:STOP
```

```
100 REM  *********************
    *
110 REM  ***                **
    *
120 REM  *** DEMO CHAR-VCHAR **
    *
130 REM  ***                **
    *
140 REM  *********************
    *
1000 :
1010 :
1020 ←GRAFS,11:←CLEAR:←COL OR 1
1030 :
1040 A$="********** "
1050 :
1060 FOR Y=-100 TO 100 STEP 20
1070 ←DRAW159,-100,0,-160,Y,0
1080 ←DRAW-160,-100,0,160,Y,0
1090 NEXT
1100 :
1110 FOR I=0 TO 3
1120 ←CHAR12+I*2,1+I*2,I,A$+"CHA
     R "
1130 ←VCHAR12+I*2,1+I*2,I,A$+"VC
     HAR "
1140 NEXT
1150 STOP
```

Lente d'ingrandimento

Con questa istruzione, frutto della creatività di Fabio Sorgato, si completa il set di comandi grafici. Tale routine simula il posizionamento di una lente d'ingrandimento sulla pagina grafica in alta risoluzione, consentendo di vedere ingrandito tutto ciò che vi appare.

Rispetto alla versione apparsa su Commodore Computer Club N.28, sono state apportate alcune variazioni allo scopo di ovviare ad un malfunzionamento che si verificava in caso di valore illegale della coordinata Y e del colore (un banale errore di salto di due istruzioni BNE), e per ottenere la compatibilità completa con tutte le altre nuove istruzioni.

Inoltre la routine è stata suddivisa in due parti per sfruttare al meglio lo spazio disponibile, così da potere disporre di più blocchi liberi per gli sprite.

Ecco quindi come viene modificata, in linea di massima, la mappa di memoria con l'inserimento dell'istruzione LENS:

```
....INDIRIZZI
C41D----------------
....PAROLE
C801----------------
....
C806----------------
....LENS (1ma parte)
C8BO----------------
....
....
CA73----------------
....LENS (2da Parte)
CAFF----------------
CB00----------------
....Blocco utilizza-
....to da Lens
CB3E----------------
....
CB40----------------
....Blocco utilizza-
....to da Lens
CB7E----------------
```

Come potete vedere non rimane molto spazio a disposizione (da $CB80 a $CBFF: due soli blocchi da 64 byte) per i vostri sprite. Perciò sta a voi decidere se e quando conviene aggiungere il comando.

L'istruzione Lens

Sintassi: LENS X,Y,C
funzione: ingrandisce la porzione di pagina grafica sottostante alla lente.

Il parametro X indica la coordinata orizzontale del centro della lente (asse ottico). I valori assegnabili sono compresi tra 0 e 511. Il valore per porre l'asse ottico al centro dello schermo e ' 148.

Il parametro Y specifica la coordinata verticale dell'asse ottico della lente. I valori possono variare tra 0 e 255. Con 89 si pone la lente al centro dello schermo.

Attenzione perchè in questo comando l'origine delle coordinate X e Y viene stabilita in alto a sinistra (e non al centro dello schermo), inoltre la Y assume valori crescenti al di sotto dell'origine. Per avere quindi una corrispondenza tra i punti disegnati e la lente occorre ricorrere alle sguenti formule:

LX=148+PX
LY=89-PY

in cui LX ed LY sono le coordinate della lente e PX e PY quelle del punto disegnato (la coordinata Z non viene presa in considerazione in quanto la lente non lavora in 3D).

Il parametro C indica il colore dei punti ingranditi e può variare tra 0 e 255: si consiglia di usare lo stesso colore scelto per il disegno. La tabella dei colori è quella solita degli sprite.

La lente è stata creata per funzionare con grafica monocromatica in alta risoluzione, quindi si consiglia di non usarla in modo multicolor o in modo testo.

Anche questo comando sfrutta l'opzione "$" per i parametri precedentemente definiti: fate quindi attenzione ad usarla con criterio se intendete avvalervene (rileggete il paragrafo a questa dedicato).

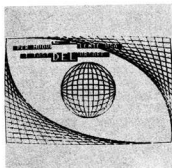
Per disattivare la lente basta eseguire:

POKE 53269,0

oppure la si può fare scomparire assegnando alla X valori che la portino oltre lo schermo visibile (es. LENS 400,0,0).

Come funziona Lens

La lente è uno sprite, in cui vengono trasferiti i dati, con le coordinate X e Y raddoppiate, posto nella zona di schermo da cui sono prelevati i dati stessi.

Il primo problema che si presenta è quello di individuare la zona di memoria da cui devono essere prelevati i dati e si risolve tramite la segunte formula:

L=START+40*(Y AND 504)+Y AND 7+X AND 504

dove L è la locazione di memoria da cui è prelevato il byte, mentre START è la locazione d'inizio della pagina grafica.

Il secondo problema si presenta nel trasferire i dati nello sprite, data la diversa configurazione di questo rispetto alla pagina grafica. Per risolverlo si deve ricorrere a "rotazioni" binarie che ogni buon programmatore esperto in linguaggio macchina dovrebbe sapere usare.

L'ultimo problema è dato dalla trasparenza dello sprite per cui ponendolo (espanso in entrambe le dimensioni) nel punto desiderato, osserveremo una sovrapposizione di punti con il disegno sottostante.

La soluzione consiste nel porre nello stesso punto un secondo sprite "pieno", cioè completamente colorato con il colore del fondo dello schermo grafico e con precedenza di visualizzazione inferiore rispetto a quello contenente i dati della lente (in questa routine vengono utilizzati lo sprite 0 per i dati e lo sprite 1 per lo sfondo).

Anche in questo caso le istruzioni da seguire per l'attivazione del nuovo comando sono:

1: Caricare il programma Basic delle routine grafiche finora realizzato.

2: Digitare il programma "Lente d'ingrandimento", registrare e verificare il tutto.

3: Dare il RUN.

Potete ora cancellare la riga 20350 che, comunque, viene neutralizzata dalla 25440.

```
24800 REM ******************
24810 REM *                *
24820 REM *    L E N T E   *
24830 REM * D'INGRANDIMENTO *
24840 REM *   BY F.SORGATO  *
24850 REM * MODIFICATA PER  *
24860 REM *    OTTENERE     *
24870 REM * COMPATIBILITA'  *
24880 REM * CON GLI  ALTRI  *
24890 REM * NUOVI COMANDI   *
24900 REM ******************
24910 REM * INDIRIZZO:      *
24920 REM * LENS=51206      *
24930 REM ******************
24940 REM * E'  NECESSARIO  *
24950 REM * SIANO GIA' STATI*
24960 REM * CARICATI TUTTI  *
24970 REM *GLI ALTRI COMANDI*
24980 REM ******************
24990 :
25000 B=0:FOR I=51206 TO 51376:RE
      AD A:B=B+A:POKE I,A:NEXT
25010 IF B<>20775 THEN PRINT"ERRO
      RE NEI DATA DEL BLOCCO 1":E
      ND
25020 B=0:FOR I=51827 TO 51967:RE
      AD A:B=B+A:POKE I,A:NEXT
25030 IF B<>18792 THEN PRINT"ERRO
      RE NEI DATA DEL BLOCCO 2":E
      ND
25040 B=0:FOR I=51199 TO 51201:RE
      AD A:B=B+A:POKE I,A:NEXT
25050 FOR I=50204 TO 50205:READ A
      :B=B+A:POKE I,A:NEXT
25060 IF B<>486 THEN PRINT"ERRORE
       NEI DATA PAROLA/INDIRIZZO"
      :END
25070 :
25080 REM    ****   ROUTINE    ****
25090 REM    ****   BLOCCO 1    ***
      *
25100 DATA  32,143,197,173,208,2,
      133,87,173,209,2,41,254,240
25110 DATA  5,162,14,108,0,3,173,
      209,2,133,88,173,210,2
25120 DATA  133,89,173,211,2,208,
      236,173,212,2,133,91,173,21
      3
25130 DATA  2,208,226,173,0,207,4
      1,15,133,90,169,44,141,248
```

```
25140 DATA  207,169,45,141,249,20
      7,169,3,141,21,208,141,23,2
      08
25150 DATA  141,29,208,165,90,141
      ,40,208,165,91,141,39,208,1
      20
25160 DATA  169,53,133,1,169,0,13
      3,94,133,2,32,115,202,160
25170 DATA  0,165,87,24,105,12,14
      4,2,160,3,141,0,208,141
25180 DATA  2,208,165,89,24,105,1
      8,141,1,208,141,3,208,165
25190 DATA  88,240,2,160,3,132,25
      1,173,16,208,41,252,5,251
25200 DATA  141,16,208,162,64,160
      ,255,189,191,3,157,255,202,
      152
25210 DATA  157,63,203,202,208,24
      3,169,55,133,1,169,96,141,6
      5
25220 DATA  3,88,96
25230 :
25240 REM   ****   BLOCCO 2    ***
      *
25250 DATA  165,89,74,74,74,10,13
      3,251,169,160,133,252,32,23
      0
25260 DATA  202,165,87,41,7,170,1
      65,87,41,248,133,90,165,89
25270 DATA  41,7,5,90,24,101,254,
      144,2,230,255,133,254,165
25280 DATA  255,24,101,88,105,224
      ,133,255,160,0,177,254,153,
      90
25290 DATA  0,165,254,24,105,7,14
      4,2,230,255,133,254,200,192
25300 DATA  4,208,235,138,240,11,
      6,93,38,92,38,91,38,90
25310 DATA  202,208,245,164,94,16
      2,0,181,90,153,192,3,230,94
25320 DATA  200,232,224,3,208,243
      ,230,89,230,2,165,2,201,21
25330 DATA  208,142,96,169,0,162,
      8,10,38,251,144,7,24,101
25340 DATA  252,144,2,230,251,202
      ,208,241,133,254,165,251,13
      3,255
25350 DATA  96
25360 :
25370 REM   ****   PAROLA    ****
25380 DATA  2,195,83
25390 :
25400 REM   ****   INDIRIZZO   **
      **
25410 DATA  6,200
25420 :
25430 REM   ** NUMERO DI COMANDI
      ATTIVATI **
25440 POKE 51051,17
```

```
100 REM   ****   DEMO LENS   ***
    *
110 :
120 ←CLEAR:←GRAF3,11:←COL OR 1
130 FOR K=0 TO 200 STEP 10
140 ←DRAW-160,100-K,0,K*1.6-160
    ,-100,0
150 ←DRAWK*1.6-161,99,0,159,99-
    K,0
160 NEXT
170 FOR K=0 TO 50 STEP 10:←CIRC
    LE0,0,0,K,50:NEXT
180 FOR K=0 TO 50 STEP 10:←CIRC
    LE0,0,0,50,K:NEXT
190 ←CHAR2,2,1," PER MUOVERE LA
     LENTE USA "
200 ←CHAR4,4,1," I TASTI DEL CU
    RSORE "
210 :
220 X=148:Y=89:S=4
230 GET A$
240 IF A$="[RIGHT]" THEN X=X+S:
    REM  CURSOR RIGHT
250 IF A$="[LEFT]" THEN X=X-S:R
    EM  CURSOR LEFT
260 IF A$="[DOWN]" THEN Y=Y+S:R
    EM  CURSOR DOWN
270 IF A$="[UP]" THEN Y=Y-S:REM
     CURSOR UP
280 ←LENSX,Y,0
290 GOTO 230
```

Come assemblare i moduli

Se non avete ancora provveduto a riunire tutti i moduli (listati Basic) in un unico programma, queste poche righe di comandi Basic vi faranno risparmiar tempo e fastidi: alla fine riunirete in un unico file "compatto" (solo 12 blocchi su disco!) i diciassette i comandi presentati.

Le operazioni da compiere sono semplici:

1: Caricare i vari moduli uno alla volta (nell'ordine prescritto dai listati) e dare il RUN dopo ciascun caricamento; oppure caricate il programma ottenuto dalla "somma" dei vari programmi delle pagine precedenti (tranne, ovviamente, i "Demo") e date il RUN.

2: Eseguire
POKE44,40:
POKE10240,0:NEW

3: Caricare il listato "Compattatore" e dare il RUN.

4: Quando compare la scritta "OK PUOI SALVARE" registrate su disco o nastro il file compatto così creato col nome che preferite.

Avvertenza: se provate a listare vedrete solo la linea:
0 SYS 2061
dal momento che il resto del programma è "invisibile" per l'interprete Basic.

Per utilizzare tale file basterà, in seguito, caricarlo come un normale programma e dare il RUN. In una frazione di secondo le nuove istruzioni saranno abilitate!

```
10 REM ****************
20 REM *"ASSEMBLATORE"*
25 REM *   DEI MODULI  *
30 REM * DELLE ROUTINE*
35 REM *   GRAFICHE    *
40 REM ****************
60 REM * PRIMA DI CA- *
62 REM * RICARE QUESTO*
63 REM * PROGRAMMA    *
65 REM * ESEGUIRE:    *
66 REM ****************
67 REM * POKE 44,40   *
70 REM * POKE 10240,0 *
72 REM * NEW          *
75 REM ****************
90 :
100 PRINTCHR$(147);SPC(252)"ATT
    ENDERE PREGO"
110 :
120 FOR I=2048 TO 2100:READ A:B
    =B+A:POKE I,A:NEXT
130 IF B<>6877 THEN PRINT"ERROR
    E NEI DATA":END
140 :
150 FOR I=0 TO 11*256:POKE 2112
    +I,PEEK(49152+I):NEXT
160 :
170 POKE 43,1:POKE 44,8:POKE 45
    ,64:POKE 46,19
180 PRINTCHR$(147);SPC(252)"OK
    PUOI SALVARE"
190 :
200 DATA  0,11,8,0,0,158,50,48,
    54,49,0,0,0
210 DATA  169,64,133,247,169,8,
    133,248,169,0
220 DATA  133,249,169,192,133,2
    50,162,11,160,0
230 DATA  177,247,145,249,200,2
    08,249,230,248,230
240 DATA  250,202,208,240,32,56
    ,199,76,60,194
```

Spostamento del punto di vista

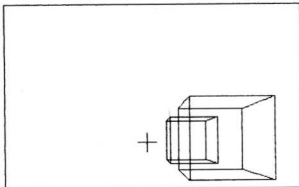
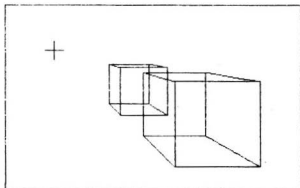
Una volta presentate le possibilità esplicite di queste collaudate routine grafiche, è giunto il momento di svelare un piccolo segreto che consentirà di ottenere nuove prestazioni.

Quando Toma scrisse le routine, pose il punto di vista al centro dello schermo perchè in questo modo le immagini tridimensionali risultano più "naturali" all'osservatore che abbia in mente, come paragone, la prospettiva fotografica.

E' però possibile, tramite due semplici POKE, disporre di altri "punti di vista" così da potere osservare gli oggetti creati da diverse angolazioni.

Ma vediamo che cosa significa spostare il punto di vista.

Immaginate di essere davanti ad u-

na finestra e di osservare un oggetto posto al di là di questa. Poniamo che sia esattamente di fronte a voi, ed immobile.

Provate ora a spostarvi lateralmente continuando a fissare l'oggetto. Facendo ciò vi accorgete di due fatti:

- La porzione della figura a voi visibile cambia: se prima vedevate solo la parte frontale, a mano a mano che vi spostate, riuscite a vedere parti laterali che vi erano prima invisibili.
- L'oggetto sembra spostarsi nella vostra stessa direzione se mantenete come "cornice" di riferimento la finestra.

La traslazione apparente è tanto maggiore quanto più l'oggetto è lontano dalla finestra e se vi spostate troppo esce addirittura dal campo visibile.

Naturalmente lo stesso discorso vale anche per traslazioni verticali del nostro occhio.

Con il computer possiamo ottenere gli stessi risultati.

Nel nostro caso lo schermo del monitor sostituisce la finestra e il punto di vista delle routine rappresenta il nostro occhio.

Come esempio potete osservare le figure A e B (ottenute direttamente dallo schermo con la routine di hard copy per la MPS 802).

La crocetta serve solo per individuare la posizione del punto di vista dal quale vengono osservati i cubi.

Chiariamo subito che la distanza dallo schermo del punto di vista, cioè la sua coordinata Z, è fissa mentre ciò che possiamo modificare sono i parametri X e Y.

Per traslare il punto lungo l'asse delle ordinate, cioè verticalmente, basta eseguire:

POKE 50151,A

dove A può assumere valori compresi tra 0 e 255.

Il valore di default delle routine è 100. Assegnando ad A il valore nullo, il punto di vista si troverà nella posizione più bassa possibile dello schermo.

Aumentando il valore avremo una corrispondente elevazione del punto di vista che, per valori superiori a 199, uscirà dallo schermo visibile.

Ciò, badate bene, non significa affatto che non vedrete più nulla, ma semplicemente che lo vedrete ancora più dall'alto.

L'operazione per traslare il punto d'osservazione lungo l'asse delle ascisse (orizzontalmente) è analoga alla precedente:

POKE 50147,A

Anche qui A può assumere valori compresi tra 0 e 255.

Il valore "normale" delle routine è 160.

Come avrete intuito, ciò comporta il fatto che il nostro punto di vista non potrà essere spostato fino al limite destro dello schermo visibile (vedi disegno 1).

Altra limitazione è data dal fatto che "Pokando" il valore zero nella locazione 50147 si entra in un caso speciale. Con tale valore, infatti, l'ascissa del nostro punto d'osservazione assume automaticamente il valore dell'ordinata. Ad esempio: se nella locazione 50151 abbiamo depositato il valore 80 e nella locazione 50147 il valore 0, le coordinate del nostro punto di vista saranno (80,80).

L'unico caso in cui potremo avere il punto con ascissa 0 si avrà quando anche l'ordinata sarà 0: punto (0,0).

Attenzione

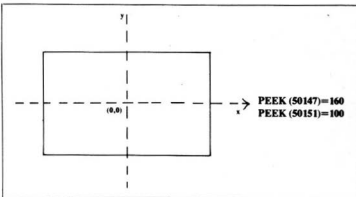
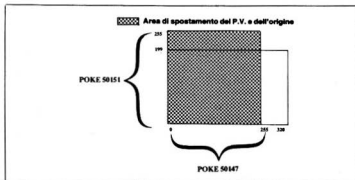
I valori delle locazioni 50151 e 50147 rimangono "fissi" e non vengono riinizializzati con un semplice RUN. Tutti i disegni realizzati dopo l'alterazione delle due locazioni saranno tracciati avendo il loro contenuto come riferimento.

Per tornare a valori "normali" inserite i due comandi:

POKE 50151,100
POKE 50147,160

Area di spostamento del P.V. e dell'origine

POKE 50151

POKE 50147

mente 160 e 100 a OX e OY fisseremo l'origine degli assi al centro dello schermo, indipendentemente dalla posizione del punto di vista.

Naturalmente le espressioni OX-PEEK(50147) e OY-PEEK(50151), che non cambiano di valore finchè non muta il contenuto delle locazioni "peekate", potete assegnarle a due variabili per evitare di ripeterle ad ogni istruzione grafica. Ad esempio:

```
DX=OX-PEEK(50147)
DY=OY-PEEK(50151)
DRAW X1+DX,Y1+DY,Z1,X2+
DX,Y2+DY,Z2
```

Demo

Per aiutarvi a comprendere meglio quanto esposto proponiamo un semplice programma demo.

Caricatelo e date il RUN (ovviamente dovrete avere prima caricato le routine grafiche).

Tramite i tasti del cursore potrete scegliere il punto da cui osservare due cubi che rimangono fissi nello spazio, anche se a voi sembrerà che si spostino in una certa misura (rileggetevi il paragrafo in cui è riportato l'esempio della finestra).

Una piccola croce individua il punto di vista. Se eleverete di molto il punto di osservazione la croce sparirà, cioè uscirà dallo schermo.

Come potete vedere (linea 230) DX e DY vengono sommati alle coordinate di uno solo dei vertici dei cubi (quello che appare più vicino, in basso a sinistra) poichè tutti gli altri sono automaticamente calcolati in funzione del primo.

Le dimensioni dei cubi sono date dalla variabile L=lato (linea 140).

Per evidenziare il punto di vista vengono tracciate due linee perpendicolari tra loro (linee 260 e 270) che si incrociano in corrispondenza dell'origine. Come potete osservare in questo caso non vengono aggiunte le rettifiche DX e DY perchè interessa ora seguire gli spostamenti del punto di vista.

Due esempi di ciò che apparirà sullo schermo li avete già visti nelle figure A e B.

Origine degli assi

Per motivi di semplificazione nel progetto delle routine, il punto di origine degli assi viene automaticamente posto, sullo schermo, in corrispondenza del punto di vista.

Ciò significa che spostando quest'ultimo si ha un'identica traslazione del primo. Se ad esempio nelle locazioni 50147 e 50151 deponete il valore zero, non avrete solo il punto di vista che si sposta nell'angolo in basso a sinistra dello schermo, ma anche l'origine degli assi. Così il punto di coordinate (0,0), che "normalmente" si trova al centro dello schermo, si sposta nel citato angolo. Per una più chiara comprensione osservate le figure A e B.

Questo movimento simultaneo, però, impedisce il cambiamento dell'angolo di osservazione degli oggetti disegnati.

Quando, infatti, per avere una diversa visuale delle figure, spostiamo il punto di vista, queste si spostano solidalmente ad esso poichè la loro posizione nello spazio (o meglio nello schermo) prende come punto di riferimento l'origine degli assi che, come già detto, corrisponde sullo schermo alla posizione del punto di vista.

Per capirci meglio fate conto di trovarvi di fronte ad uno specchio: per quanto vi spostiate lateralmente, la vostra immagine vi seguirà solidalmente e non riuscirete mai a vedervi di... profilo.

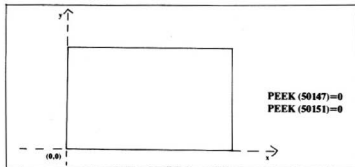
Il rimedio per mantenere fissi gli oggetti rispetto allo schermo, e quindi visibili sotto diverse angolazioni, è semplice.

Basta correggere i valori delle coordinate X e Y nel modo indicato:

```
X=X+OX-PEEK(50147)
Y=Y+OY-PEEK(50151)
```

Le variabili OX e OY possono assumere valori tra 0 e 255 ed indicano la posizione FISSA dello schermo che vogliamo sia presa come punto di riferimento (cioè come origine) dalle coordinate.

Ad esempio assegnando rispettiva-

PEEK (50147)=0
PEEK (50151)=0

```
100 REM *********************
    **********
105 REM ***  ESEMPIO DI SPOSTA
    MENTO ***
110 REM *** DEL PUNTO DI VISTA
     DELLE ***
115 REM ***   ROUTINES GRAFICH
    E II    ***
120 REM *********************
    **********
130 :
140 H=150:K=153:L=110:←GRAF11,1
    :←COL OR 1
150 ←CLEAR
160 :
170 REM *** ABILITAZIONE NUOVO
     PUNTO DI VISTA ***
180 POKE 50147,K:POKE 50151,H
190 :
200 REM *** RETTIFICA DELLE CO
    ORDINATE PER MANTENERE ***
210 REM *** L'ORIGINE DEGLI AS
    SI AL CENTRO DELLO SCHERMO
    ***
220 DX=160-PEEK(50147):DY=100-P
    EEK(50151)
230 X=DX+49:Y=DY-100: REM ** C
    OORD. DI UN VERTICE DEI CUB
    I **
240 :
250 REM ***  EVIDENZIA IL PUNT
    O DI VISTA  ***
260 ←DRAW-10,0,0,10,0,0
270 ←DRAW0,-10,0,0,10,0
280 :
290 REM ***  DISEGNA I CUBI
    ***
300 FOR Z=50 TO 300 STEP 250:FO
    R I=0 TO 1
310 ←DRAWX,Y+I*L,Z,X+L,Y+I*L,Z
320 ←DRAW$,$,Z+L,$,$,Z+L
330 ←DRAWX+L,$,Z,$,$,$
340 ←DRAWX,$,Z,X,$,$
350 NEXT
360 FOR I=0 TO 1
370 ←DRAWX,Y,Z+I*L,X,Y+L,Z+I*L
380 ←DRAWX+L,$,$,X+L,$,$
390 NEXT:NEXT
400 :
410 REM ***  SCELTA DEL NUOVO
    PUNTO DI VISTA  ***
420 GET A$:IF A$="" THEN 420
430 IF A$=CHR$(145) AND H<205 T
    HEN H=H+50
440 IF A$=CHR$(17) AND H>49 THE
    N H=H-50
450 IF A$=CHR$(29) AND K<205 TH
    EN K=K+50
460 IF A$=CHR$(157) AND K>49 TH
    EN K=K-50
470 GOTO 150
```

Hard Copy della pagina grafica su MPS-802

Tutti i possessori della vecchia, ma ottima, stampante MPS 802 hanno prima o poi desiderato, magari utilizzando il Simons' Basic o altri programmi dotati dell'opzione "hard copy", di riportare il contenuto della pagina grafica su carta.

Purtoppo il risultato di una simile richiesta è quasi sempre un inchiodamento della stampante con la spia rossa che lampeggia minacciosamente.

La gestione grafica di questa stampante è infatti differente da quella utilizzata dalle altre stampanti Commodore.

Questa incompatibilità può essere davvero seccante soprattutto perchè "tarpa le ali" ad una macchina dalle prestazioni superiori (rispetto alle MPS 801-803) in fatto di qualità di stampa.

Disperarsi comunque non serve: è molto meglio copiare questa breve routine e inserirla nei propri programmi al posto di quelle "802-incompatibili".

La gestione della grafica MPS/802

Come saprete, se conoscete l'inglese e se avete letto il manuale, è possibile definire e comunicare alla stampante un carattere, che non sia tra quelli del normale set di caratteri, e farlo stampare.

Ogni carattere è il risultato di una matrice di 8x8 punti e le informazioni circa i punti della matrice da scrivere, o da lasciare in bianco, vengono prese da otto byte (che come è noto sono formati ognuno da otto bit) da comunicare tramite un Print CHR$ alla stampante, utilizzando un apposito canale (indirizzo secondario 5).

Il manuale spiega chiaramente come calcolare i dati da comunicare, perciò non ci dilunghiamo.

A questo punto avrete già intuito la tecnica da utilizzare per copiare la pagina grafica (che è un insieme ordinato di byte): basta passarne alla stampante un "pezzetto" di 8x8 punti per volta e farlo stampare come se fosse un carattere ridefinito.

Senonchè sullo schermo i byte sono posti "orizzontalmente" mentre la stampante li butta fuori "verticalmente". Occorre perciò una routine che si occupi di trasformare gli otto

byte da comunicare in modo che l'uscita su carta sia corretta. E' un'operazione che riesce molto più facilmente in Linguaggio Macchina (utilizzando le istruzioni di scorrimento e di rotazione) che in Basic.

A questo punto sembrerebbe tutto risolto, ma non abbiamo fatto i conti con la diabolicità dei progettisti della Commodore. Nella nota in fondo a pagina 24 del manuale, infatti, viene detto che si possono avere caratteri programmabili diversi, sulla stessa linea di stampa, ma solo "by overprinting".

Che cosa significa? Finchè non mandate alla stampante un segnale di ritorno carrello, CHR$(13), non potete cambiare il carattere grafico da stampare! Ma noi dobbiamo stamparne 40 di caratteri diversi per ogni riga.

La soluzione esiste, anche se piuttosto macchinosa: basta riscrivere la stessa linea 40 volte senza fare avanzare la carta ("overprinting"), "spostando" ogni volta la posizione in cui stampare il carattere (diverso ogni volta) di una posizione verso destra.

Il programma che svolge tale compito è più semplice della spiegazione precedente anche se non brilla certo per velocità di esecuzione: per stampare l'intera pagina grafica occorrono più di cinque minuti (tenete conto che vengono stampate 1000 linee "virtuali").

Mediante l'apertura di un altro degli undici canali di comunicazione, e precisamente il n.6, si può stabilire, entro certi limiti, quanto deve avanzare la carta tra una linea e la successiva. Un esempio di come utilizzare questa opzione si trova a pagina 25 del manuale.

Inviando alla stampante, tramite questo canale, un CHR$(0) otterremo di non fare avanzare la carta quando si ha il "ritorno carrello".

Con CHR$(20), invece, le linee di stampa vengono poste una sotto l'altra senza lasciare spazi.

Routine di Hard Copy

Commentiamo, dunque, il breve listato proposto.

Come molti sapranno è possibile sistemare la pagina grafica in varie posizioni nella memoria.

La routine proposta stampa la pagina posizionata a partire dalla locazione 57344 (cioè quella utilizzata dalle routine grafiche qui presentate).

Basta comunque cambiare il sesto DATA (224) della linea 10, che indica la locazione iniziale della pagina grafica, per fare stampare qualsiasi pagina desideriate.

Il valore da inserire va calcolato nel modo seguente:

DATA=I/256

dove I è l'indirizzo in cui inizia la pagina grafica.

Ad esempio il valore da sostituire a 224 sarà 160 per la pagina posizionata "sotto" l'interprete Basic (40960/256=160).

Può interessare sapere che il Simons' Basic utilizza la nostra stessa pagina.

La routine in Linguaggio Macchina viene allocata nel buffer del registratore; ne consegue che (per sicurezza) è bene riallocarla (cioè fare eseguire la linea 1000) ogni volta che utiizzate questo breve programma.

Poichè il comando RESTORE fa iniziare la lettura dal primo DATA in memoria, assicuratevi che le linee che contengono i DATA proposti siano le prime linee di DATA dei programmi in cui includete tale routine.

Un'ulteriore informazione sembra doverosa: oltre alle locazioni dalla 820 alla 891 vengono utilizzate dalla routine in linguaggio macchina, per le "manipolazioni" dei dati da trasmettere, anche le locazioni dalla 892 alla 907.

Due considerazioni riguardo al formato di stampa.

La prima è che una schermata viene ridotta ad un disegno di 9.5*6 cm. (la scala è comunque 1:1). Potete ottenere un disegno largo il doppio inserendo, nella linea 1060, un CHR$ (14):

```
1060 PRINT#4,CHR$(14)B$;:
B$=B$+CHR$(32)
```

però occorreranno ben 23 minuti per il completamento della stampa.

Seconda considerazione: la variabile B$, introdotta nella linea 1030, determina a quale distanza dal margine sinistro del foglio deve essere stampata la pagina grafica. La distanza viene determinata dal numero di SPAZI (CHR$(32)) che questa variabile contiene e ciò lo decidete voi rispondendo alla domanda che il programma pone all'inizio.

E veniamo all'ultima linea del listato.

Questa serve per resettare la stampante, una volta terminata l'operazione di hard copy. E' necessaria perchè la stampante torni a scrivere con le linee spaziate tra loro "normalmente". In teoria basterebbe comunicare, attraverso il canale 6, il valore di "default" dell'interlinea (che a pagina 25 del manuale viene scritto essere 24) ma una prova in tale senso ha dato linee con spaziatura inferiore al normale. A "occhio" il valore corretto dovrebbe essere 34 ma, a scanso d'errori, è preferibile un brutale Reset software che aggiusta tutto.

IMPORTANTE: quando vengono premuti i tasti Run/Stop e Restore contemporaneamente la pagina grafica viene "sporcata" (come già accennato nelle spiegazioni delle routine grafiche). State quindi attenti a non effettuare tale operazione prima di avere stampato il disegno.

Ricordiamo che al momento dell'esecuzione della routine di hard copy non occorre che sia visualizzata sul monitor la pagina grafica.

Riassumiamo infine le operazioni da effettuare per stampare la pagina grafica.

• Disegnare qualcosa nella pagina grafica.
• Caricare il programma "MPS/802 Hard Copy".
• Dare il RUN.

Ovviamente potete unire la routine di hard copy e quella che consente di disegnare in un unico programma, senza doverla caricare ogni volta che volete stampare.

```
   1 REM *********************
   3 REM * MPS 802 HARD COPY *
   4 REM *                   *
   5 REM * BY DANILO TOMA    *
   7 REM *********************
   8 :
  10 DATA  169,0,133,253,169,224
     ,133,254,96,160
  20 DATA  7,32,108,3,177,253,15
     3,124,3,136
  30 DATA  16,248,32,116,3,160,7
     ,162,7,94
  40 DATA  124,3,106,202,16,249,
     153,132,3,136
  50 DATA  16,241,24,165,253,105
     ,8,133,253,165
  60 DATA  254,105,0,133,254,96,
     120,169,252,37
  70 DATA  1,133,1,96,169,3,5,1,
     133,1
  80 DATA  88,96
1000 RESTORE :K=0:FOR I=820 TO 8
     91:READ A:K=K+A:POKE I,A:NE
     XT
1002 IF K<>8066 THEN PRINT"ERROR
     E NEI DATA":END
1005 INPUT "DISTANZA DAL MARGINE
      SINISTRO (0-40)";C$
1006 C=VAL(C$):IF C<0 OR C>40 TH
     EN 1005
1007 C$="":IF C=0 THEN 1010
1008 FOR I=1 TO C:C$=C$+CHR$(32)
     :NEXT
1009 :
1010 OPEN 4,4:OPEN 5,4,5:OPEN 6,
     4,6:SYS820
1020 FOR K=0 TO 24:PRINT#6,CHR$(
     0)
1030 B$=C$:FOR J=0 TO 39:SYS829
1040 A$="":FOR I=900 TO 907:A$=A
     $+CHR$(PEEK(I)):NEXT
1050 PRINT#5,A$
1060 PRINT#4,B$;:B$=B$+CHR$(32)
1070 PRINT#4,CHR$(254):NEXT
1080 PRINT#6,CHR$(20):PRINT#4:NE
     XT
1090 CLOSE 4:CLOSE 5:CLOSE 6
1100 OPEN 4,4,10:PRINT#4:CLOSE 4
```

Hard Copy della pagina grafica su MPS-803

Questa routine ricorre, per il funzionamento, a raffinate tecniche di programmazione in LM (vedi disassemblato commentato): darne una spiegazione dettagliata sarebbe fuori luogo.

Ci limiteremo a ricordare alcuni punti fondamentali per il suo corretto utilizzo.

• La routine LM occupa le locazioni da 40704 a 40940 utilizzate normalmente dal Basic per allocare le stringhe.

• Onde evitare pericolose cancellazioni è necessario SEMPRE inserire i comandi che alterano le locazioni del Top di memoria (vedi riga 0).

• Dopo aver caricato (e lanciato) le routine grafiche, caricare e lanciare il semplice programma Basic pubblicato: automaticamente i puntatori verranno messi a posto.

• Caricare e/o digitare i programmi che fanno ricorso alle routine grafiche e impartire SYS 40704 tutte le volte che si intende inviare alla stampante la pagina grafica.

```
    0 POKE 56,159:POKE 55,0:CLR :
      REM  SPOSTA PUNTATORI TOP M
      EMORY
    5 :
   10 REM HARD COPY DI PAGINA
   15 REM GRAFICA PER STAMPANTI
   20 REM MPS/801, /803 E COMPATI
      BILI
   30 REM BY FABIO SORGATO
   35 :
   40 REM COPY: SYS 40704
   50 REM PER INTERROMPERE: RUN/S
      TOP
   60 REM
  100 FOR K=40704 TO 40940:READ A
      :POKE K,A:B=B+A:NEXT
  110 IF B=32242 THEN PRINT"ERROR
      E NEI DATI!"
  120 END
60000 DATA 076,137,159,032,031
60010 DATA 159,169,096,162,004
60020 DATA 160,000,032,186,255
60030 DATA 169,000,032,189,255
60040 DATA 032,192,255,162,096
60050 DATA 076,201,255,032,181
60060 DATA 171,169,096,076,195
60070 DATA 255,169,000,141,236
60080 DATA 159,165,088,240,015
60090 DATA 201,001,240,005,238
60100 DATA 236,159,024,096,165
60110 DATA 087,201,064,176,245
60120 DATA 165,079,208,241,165
```

DISASSEMBLATO COMMENTATO

```
COMPLEMENTA A 2

C000 EOR #$FF      ;ESEGUE IL COMPLEMENTO
C002 STA $FA,X     ;A DUE DEL NUMERO
C004 LDA $F9,X     ;CONTENUTO IN A
C006 EOR #$FF      ;ED IN $FA
C008 STA $F9,X     ;RISULTATO IN $F9-FA+X
C00A INC $F9,X     ;
C00C BNE $C010     ;
C00E INC $FA,X     ;
C010 LDY #$CA      ;CARICA Y CON COD. DEX
C012 LDX #$FF      ;E X CON $FF
C014 RTS           ;

TRASFERISCE COORDINATE

C015 LDX #$03      ;TRASFERISCE I VALORI
C017 LDA $02D6,X ;DEI PARAMETRI Y2 E X2
C01A STA $5D,X     ;DAL "MAGAZZINO"
                   ;ALL'AREA DI LAVORO
C01C DEX           ;
C01D BPL $C017     ;
C01F LDA $02D0     ;IDEM PER X1
C022 STA $57       ;
C024 LDA $02D1     ;
C027 STA $58       ;
C029 LDA $02D2     ;IDEM PER Y1
C02C STA $5B       ;
C02E STA $4E       ;
C030 LDA $02D3     ;
C033 STA $5C       ;
C035 STA $4F       ;
C037 RTS           ;
C038 NOP           ;
C039 NOP           ;

TABELLA PER PLOT/UNPLOT MULTICOLOR

C03A C0 30 0C 03 00 55 AA FF

C042 NOP           ;
C043 NOP           ;
C044 NOP           ;
C045 NOP           ;
C046 NOP           ;
C047 NOP           ;


ROUTINE PRINCIPALE DI DRAW

C048 LDY #$E8      ;CARICA Y CON COD. INX
C04A LDX #$00      ;E X CON $00
C04C SEC           ;CALCOLA DX=X2-X1
C04D LDA $5D       ;CIOE' LA DIFFERENZA
C04F SBC $57       ;TRA LE ASCISSE DEGLI
C051 STA $F9       ;ESTREMI DELLA LINEA
C053 LDA $5E       ;
C055 SBC $58       ;
C057 STA $FA       ;
C059 BPL $C05E     ;SE DX E' POSIT. SALTA
C05B JSR $C000     ;COMPLEMENTA IL RISULT.
C05E STY $C137     ;MODIFICA LA ROUTINE
C061 STX $C13B     ;IN ACCORDO CON DX
C064 LDY #$C6      ;CARICA Y CON COD. DEC$
C066 TXA           ;
C067 BNE $C06B     ;SE DX NEGATIVO SALTA
C069 LDY #$E6      ;CARICA Y CON COD. INC$
C06B STY $C13E     ;
C06E LDY #$E8      ;CARICA Y CON COD. INX
C070 LDX #$00      ;E X CON $00
C072 SEC           ;CALCOLA DY=Y2-Y1
C073 LDA $5F       ;CIOE' LA DIFFERENZA
C075 SBC $5B       ;TRA LE ORDINATE DEGLI
C077 STA $FD       ;ESTREMI DELLA LINEA
C079 LDA $60       ;
C07B SBC $5C       ;
C07D STA $FE       ;
C07F BPL $C086     ;SE DY POSITIVO SALTA
C081 LDX #$04      ;COMPLEMENTA IL RISULT.
C083 JSR $C000     ;MODIFICA LA ROUTINE
C086 STY $C16A     ;IN ACCORDO CON DY
C089 STX $C16E     ;
C08C STX $59       ;
C08E STX $5A       ;
C090 LDY #$C6      ;CARICA Y CON COD. DEC$
C092 TXA           ;
C093 BNE $C097     ;SE DY NEGATIVO SALTA
C095 LDY #$E6      ;CARICA Y CON COD. INC$
C097 STY $C171     ;
C09A LDY #$38      ;CARICA Y CON COD. SEC
C09C LDA #$E5      ;E A CON SBC$
C09E CPX #$00      ;
C0A0 BNE $C0A6     ;SALTA SE DY NEGATIVO
C0A2 LDY #$18      ;CARICA Y CON COD. CLC
C0A4 LDA #$65      ;E A CON COD. ADC$
C0A6 STY $C148     ;
C0A9 STA $C14B     ;
C0AC STA $C151     ;
C0AF STA $C157     ;
C0B2 STA $C15D     ;
C0B5 LDA #$EA      ;CARICA A CON COD. NOP
C0B7 STA $C173     ;PER ESEGUIRE LINEA
                   ;NON PARALLELA ALL'ASSE
                   ;DELLE ORDINATE
C0BA LDA $F9       ;
C0BC BNE $C0C0     ;SE DX<>0 SALTA
C0BE LDA $FA       ;
C0C0 BNE $C0DD     ;
C0C2 LDA #$60      ;CARICA A CON COD. RTS
C0C4 STA $C173     ;ED ESEGUE LINEA
C0C7 CLC           ;PARALLELA ALL'ASSE
C0C8 BCC $C0CD     ;DELLE ORDINATE
C0CA JSR $C168     ;
C0CD JSR $C18D     ;
C0D0 LDA $4E       ;CONTROLLA SE LA LINEA
C0D2 CMP $5F       ;E' STATA COMPLETATA
C0D4 BNE $C0CA     ;
C0D6 LDA $4F       ;
C0D8 CMP $60       ;
C0DA BNE $C0D4     ;
C0DC RTS           ;
C0DD INC $F9       ;CALCOLA DY/DX
C0DF BNE $C0E3     ;CIOE' IL COEFFICIENTE
C0E1 INC $FA       ;ANGOLARE DELLA LINEA
C0E3 INC $FD       ;DA TRACCIARE
C0E5 BNE $C0E9     ;
```

```
CØE7 INC $FE      ;
CØE9 LDA #$ØØ     ;ROUTINE DI DIVISIONE
CØEB STA $FB      ;
CØED STA $FC      ;
CØEF STA $F7      ;
CØF1 STA $F8      ;
CØF3 LDX #$21     ;
CØF5 LDA $F7      ;
CØF7 SEC          ;
CØF8 SBC $F9      ;
CØFA TAY          ;
CØFB LDA $F8      ;
CØFD SBC $FA      ;
CØFF BCC $C106    ;
C1Ø1 STA $F8      ;
C1Ø3 TYA          ;
C1Ø4 STA $F7      ;
C1Ø6 ROL $FB      ;RISULTATO IN $FE-FD
C1Ø8 ROL $FC      ;(PARTE INTERA)
C1ØA ROL $FD      ;E IN $FC-FB
C1ØC ROL $FE      ;(PARTE DECIMALE)
C1ØE ROL $F7      ;RESTO IN $F8-F7
C11Ø ROL $F8      ;
C112 DEX          ;
C113 BNE $CØF5    ;
C115 LDA $F7      ;
C117 BNE $C11D    ;SALTA L'ARROTONDAMENTO
C119 LDA $F8      ;PER ECCESSO
C11B BEQ $C124    ;SE NON C'E' RESTO
C11D LDX #$ØØ     ;
C11F INX          ;
C12Ø INC $FA,X    ;ARROTONDAMENTO
C122 BEQ $C11F    ;DEL RISULTATO
C124 LDY #$Ø1     ;
C126 LDA $FD      ;SE COEFF. ANG. >=1
C128 BNE $C13Ø    ;METTE FLAG =1
C12A LDA $FE      ;
C12C BNE $C13Ø    ;
C12E LDY #$ØØ     ;ALTRIMENTI FLAG=Ø
C13Ø STY $5Ø      ;
C132 CLC          ;INIZIA ESECUZIONE
C133 BCC $C14Ø    ;
C135 LDX $57      ;INCREM. O DECREM.
C137 DEX          ;ASCISSA IN ACCORDO CON
C138 STX $57      ;DX
C13A CPX #$FF     ;I CODICI DEX E DEC $58
C13C BNE $C14Ø    ;POSSONO INFATTI
C13E DEC $58      ;DIVENTARE INX INC $58
                  ;IN BASE AL SEGNO DI DX
                  ;(VEDI $CØ48-CØ68)
C14Ø LDA $5B      ;
C142 STA $4E      ;CONSERVA VALORE
C144 LDA $5C      ;ORDINATA ATTUALE
C146 STA $4F      ;
C148 CLC          ;INCREMENTA O DECREM.
C149 LDA $59      ;ORDINATA TRAMITE IL
C14B ADC $FB      ;COEFF. ANG.
C14D STA $59      ;I VARI ADC$ DIVERREBBE-
C14F LDA $5A      ;RO SBC$ SE DY<Ø
C151 ADC $FC      ;(VEDI $CØ6E-CØB7)
C153 STA $5A      ;
C155 LDA $5B      ;
C157 ADC $FD      ;
C159 STA $5B      ;
C15B LDA $5C      ;
C15D ADC $FE      ;
C15F STA $5C      ;
C161 JSR $C18D    ;PLOTTA IL PUNTO
C164 LDA $5Ø      ;CONTROLLA FLAG
C166 BEQ $C18Ø    ;SE COEFF. ANG.<1 SALTA
C168 LDX $4E      ;
C16A INX          ;INCREMENTA O DECREM.
C16B STX $4E      ;(STESSE CONSIDERAZIONI
C16D CPX #$ØØ     ;PRECEDENTI) L'ORDINATA
C16F BNE $C173    ;FINCHE' NON VIENE
C171 INC $4F      ;RIEMPITO IL VUOTO TRA
C173 NOP          ;DUE PUNTI DI ASCISSA
C174 LDA $4E      ;DIFFERENTE
C176 CMP $5B      ;
C178 BNE $C161    ;
C17A LDA $4F      ;RIPETI FINCHE' LA
C17C CMP $5C      ;LINEA NON E'
C17E BNE $C178    ;TERMINATA
C18Ø LDA $57      ;
C182 CMP $5D      ;
C184 BNE $C135    ;
C186 LDA $58      ;
C188 CMP $5E      ;
C18A BNE $C184    ;
C18C RTS          ;
```

ROUTINE PRINCIPALE DI PLOT

```
C18D LDA $58      ;SE IL VALORE X E'
C18F BEQ $C19D    ;INFERIORE A 256
                  ;CONTROLLA LE Y
C191 CMP #$Ø1     ;SE E' SUPERIORE A 511
C193 BEQ $C196    ;ESCE
C195 RTS          ;
C196 LDA $57      ;SE E'INFERIORE A 320
C198 CMP #$4Ø     ;CONTROLLA LE Y
C19A BCC $C19D    ;ALTRIMENTI
C19C RTS          ;ESCE
C19D LDA $4F      ;SE IL VALORE Y E'
C19F BEQ $C1A2    ;SUPERIORE A 256
C1A1 RTS          ;ESCE
C1A2 LDA $4E      ;SE IL VALORE Y
C1A4 CMP #$C8     ;E' INFERIORE A 2ØØ
C1A6 BCC $C1A9    ;ESEGUE PLOT
C1A8 RTS          ;ALTRIMENTI ESCE
C1A9 LDA $4E      ;PRENDE I PRIMI 3 BIT
C1AB AND #$Ø7     ;DI Y E
C1AD STA $F7      ;LI SALVA
C1AF LDA $4E      ;PRENDE Y
C1B1 LSR ,A       ;DIVIDE PER 4
C1B2 LSR ,A       ;
C1B3 AND #$FE     ;SCARTA IL BIT Ø
C1B5 TAY          ;USA IL RISULTATO COME
C1B6 LDA $C1EB,Y  ;PUNTATORE ALLA
                  ;TABELLA DOVE PRENDE
                  ;UN BYTE BASSO
C1B9 SEC          ;A CUI SOTTRAE I PRIMI
C1BA SBC $F7      ;TRE BIT DI Y
C1BC STA $F7      ;SALVA
C1BE LDA $C1EC,Y  ;PRENDE UN BYTE ALTO
C1C1 CLC          ;DALLA TABELLA
```

```
C1C2 ADC $58       ;ADDIZIONA IL BYTE ALTO
C1C4 STA $FB       ;DI X E SALVA
C1C6 LDA $57       ;PRENDE I BIT 3-7 DI X
C1C8 AND #$F8      ;E LI METTE NEL
C1CA TAY           ;REGISTRO Y
C1CB LDA $57       ;METTE NELL'ACCUMULATO-
C1CD AND #$07      ;RE I PRIMI 3 BIT DI X
C1CF JMP $C224     ;ESEGUE PLOT MULTICOLOR
                   ;O HIRES SECONDO
                   ;I CASI
```

PLOT HIRES

```
C1CF TAX           ;QUESTE DUE ISTRUZIONI
C1D0 LDA $C1E3,X   ;SOSTITUISCONO
                   ;JMP $C224 QUANDO E'
                   ;ABILITATO IL MODO HIRES
C1D3 LDX $02       ;SE IL COLORE E' ZERO
C1D5 BEQ $C1DC     ;ESEGUE UNPLOT
C1D7 ORA ($F7),Y   ;PLOTTA IL PUNTO
C1D9 STA ($F7),Y   ;INDICATO DAI
                   ;PUNTATORI
C1DB RTS
```

UNPLOT

```
C1DC EOR #$FF      ;CANCELLA IL PUNTO
C1DE AND ($F7),Y   ;INDICATO DAI
C1E0 STA ($F7),Y   ;PUNTATORI
C1E2 RTS
```

TABELLA PER INDIVIDUAZIONE BIT

```
C1E3 80 40 20 10 08 04 02 01
```

TABELLA PER INDIVIDUAZIONE BYTE
SULLA MAPPA GRAFICA

```
C1EB 07 FE C7 FC B7 FB 47 FA
C1F3 07 F9 C7 F7 B7 F6 47 F5
C1FB 07 F4 C7 F2 B7 F1 47 F0
C203 07 EF C7 ED B7 EC 47 EB
C20B 07 EA C7 E8 B7 E7 47 E6
C213 07 E5 C7 E3 B7 E2 47 E1
C21B 07 E0

C21D NOP           ;RIEMPITIVO
C21E NOP           ;IDEM
C21F NOP           ;IDEM
C220 NOP           ;IDEM
C221 NOP           ;IDEM
C222 NOP           ;IDEM
C223 NOP           ;IDEM
```

PLOT MULTICOLOR

```
C224 LSR ,A        ;
C225 TAX           ;
C226 LDA $C03A,X   ;PRENDE COPPIA DI BIT
                   ;DA TABELLA MULTICOLOR
C229 EOR #$FF      ;SPEGNE EVENTUALI BIT
C22B AND ($F7),Y   ;GIA' ACCESI IN CORRI-
C22D STA ($F7),Y   ;SPONDENZA DEL PUNTO
                   ;DA PLOTTARE
C22F LDA $C03A,X   ;RIPRENDE COPPIA E LA
C232 LDX $02       ;MODIFICA IN ACCORDO
C234 AND $C03E,X   ;CON COLORE SCELTO
C237 ORA ($F7),Y   ;PLOTTA IL PUNTO
C239 STA ($F7),Y   ;INDICATO DAI PUNTAT.
C23B RTS
```

NEW ERRORS ROUTINE

```
C23C LDA #$47      ;AGGIORNA I PUNTATORI
C23E STA $0300     ;PER LA NUOVA ROUTINE
C241 LDA #$C2      ;DI ERRORE
C243 STA $0301     ;
C246 RTS           ;
C247 STX $FE       ;CONSERVA COD. ERRORE
C249 LDA #$20      ;
C24B AND $D011     ;SE IN MODO TESTO
C24E BEQ $C259     ;SALTA
C250 LDA #$0E      ;ALTRIMENTI TORNA AL
C252 STA $FB       ;MODO TESTO CON IL
C254 LDA #$06      ;COLORI AZZURRO PER I
C256 JSR $C6CF     ;CARATTERI E BLU PER LO
                   ;SFONDO
C259 LDX $FE       ;RIPRENDE COD. ERRORE
C25B BMI $C260     ;SALTA SE COD.>$F7
C25D JMP $A43A     ;ESEGUE ERROR ROUTINE
C260 JMP $A474     ;ESEGUE READY ROUTINE
```

*** CLEAR ***

```
C263 LDA #$E0      ;POSIZIONA PUNTATORI
C265 STA $FA       ;ALL'INIZIO DELLA
C267 LDA #$00      ;PAGINA GRAFICA
C269 STA $F9       ;
C26B LDX #$20      ;
C26D TAY           ;
C26E STA ($F9),Y   ;AZZERA TUTTI I BYTES
C270 INY           ;DELLA PAGINA GRAFICA
C271 BNE $C26E     ;
C273 INC $FA       ;
C275 DEX           ;
C276 BNE $C26E     ;
C278 RTS           ;

C279 LDA #$CC      ;POSIZIONA PUNTATORI
C27B STA $FA       ;ALL'INIZIO DELLA
C27D LDY #$00      ;MEMORIA DI SCHERMO
C27F STY $F9       ;
C281 LDA $FB       ;PRENDE COLORE DEI
C283 ASL ,A        ;PUNTI ACCESI E LO
C284 ASL ,A        ;MOLTIPLICA PER 16
C285 ASL ,A        ;
C286 ASL ,A        ;
C287 STA $FD       ;CONSERVA RISULTATO
C289 LDA $FC       ;PRENDE COLORE DEI
C28B AND #$0F      ;PUNTI SPENTI
C28D CLC           ;
C28E ADC $FD       ;SOMMA I DUE COLORI
C290 LDX #$04      ;
C292 STA ($F9),Y   ;METTE RISULTATO
C294 INY           ;IN TUTTE LE LOCAZIONI
C295 BNE $C292     ;DELLA MEMORIA
```

```
C297 INC $FA    ;DI SCHERMO
C299 DEX        ;
C29A BNE $C292  ;
C29C RTS        ;
C29D LDA $FB    ;ISTRUZ. NON UTILIZZATA
C29F BEQ $C2B7  ;IDEM
C2A1 LDA #$20   ;ABILITA MODO GRAFICO
C2A3 ORA $D011  ;
C2A6 STA $D011  ;
C2A9 LDA #$38   ;
C2AB STA $D018  ;
C2AE LDA #$FC   ;
C2B0 AND $DD00  ;
C2B3 STA $DD00  ;
C2B6 RTS        ;
C2B7 LDA #$DF   ;ABILITA MODO TESTO
C2B9 AND $D011  ;
C2BC STA $D011  ;
C2BF LDA #$15   ;
C2C1 STA $D018  ;
C2C4 LDA #$03   ;
C2C6 ORA $DD00  ;
C2C9 STA $DD00  ;
C2CC LDA $FB    ;
C2CE STA $0286  ;
C2D1 RTS        ;
```

TRASFORMA COORDINATE 3D IN 2D

```
C2EC LDA #$00   ;INIZIALIZZA FLAGS
C2EE STA $61    ;PER PLOT
C2F0 STA $63    ;
C2F2 BEQ $C346  ;ED ESEGUE TRASFORMAZ.
C2F4 LDA #$02   ;INIZIALIZZA FLAGS
C2F6 STA $61    ;PER CIRCLE/ARC
C2F8 STA $63    ;
C2FA LDA #$00   ;
C2FC STA $62    ;
C2FE BEQ $C30C  ;ED ESEGUE TRASFORMAZ.
C300 LDA #$01   ;INIZIALIZZA FLAGS
C302 STA $61    ;PER DRAW
C304 LDA #$00   ;
C306 STA $63    ;
C308 LDA #$06   ;
C30A STA $62    ;
C30C LDA $5D    ;ESEGUE TRASFORMAZIONE
C30E STA $FE    ;DI X2 PER DRAW
C310 LDA $5E    ;O DI RX (RAGGIO)
C312 STA $FF    ;PER CIRCLE/ARC
C314 LDY $62    ;FLAG
C316 LDA $02D4,Y ;PRENDE Z2 OPPURE Z1
C319 STA $F7    ;SECONDO CHE SIA IN
C31B LDA $02D5,Y ;ESECUZ. DRAW O CIRCLE
C31E STA $F8    ;
C320 LDA #$00   ;FLAG X/Y
C322 STA $51    ;SETTATO PER X
C324 JSR $C383  ;TRASFORMA 3D IN 2D
C327 LDA $FD    ;
C329 STA $5D    ;CONSERVA RISULTATO
C32B LDA $FE    ;
C32D STA $5E    ;
C32F LDA $5F    ;IDEM PER Y2 O RY
C331 STA $FE    ;
C333 LDA $60    ;
C335 STA $FF    ;
C337 LDA #$01   ;FLAG X/Y
C339 STA $51    ;SETTATO PER Y
C33B JSR $C38D  ;TRASFORMA 3D IN 2D
C33E LDA $FD    ;
C340 STA $5F    ;CONSERVA RISULTATO
C342 LDA $FE    ;
C344 STA $60    ;
C346 LDA $57    ;IDEM PER X1
C348 STA $FE    ;
C34A LDA $58    ;
C34C STA $FF    ;
C34E LDA $02D4  ;PRENDE SEMPRE Z1
C351 STA $F7    ;
C353 LDA $02D5  ;
C356 STA $F8    ;
C358 LDA #$00   ;FLAG X/Y
C35A STA $51    ;SETTATO PER X
C35C JSR $C383  ;TRASFORMA 3D IN 2D
C35F LDA $FD    ;
C361 STA $57    ;CONSERVA RISULTATO
C363 LDA $FE    ;
C365 STA $58    ;
C367 LDA $5B    ;IDEM PER Y1
C369 STA $FE    ;
C36B LDA $5C    ;
C36D STA $FF    ;
C36F LDA #$01   ;FLAG X/Y
C371 STA $51    ;SETTATO PER Y
C373 JSR $C38D  ;TRASFORMA 3D IN 2D
C376 LDA $FD    ;
C378 STA $5B    ;CONSERVA IL RISULTATO
C37A STA $4E    ;
C37C LDA $FE    ;
C37E STA $5C    ;
C380 STA $4F    ;
C382 RTS        ;
C383 LDA #$00   ;DISTANZA P.TO DI VISTA
C385 STA $F9    ;DALLO SCHERMO V=256
C387 INC $F8    ;VZ=V+Z
C389 BNE $C38D  ;
C38B INC $F9    ;
C38D LDA $FF    ;CONSERVA BYTE ALTO X/Y
C38F STA $50    ;COME FLAG POS./NEG.
C391 BPL $C398  ;SE COORDINATA NEGATIVA
C393 LDX #$05   ;COMPLEMENTA A 2
C395 JSR $C000  ;
C398 LDA #$00   ;INIZIALIZZA DIVIDENDO
C39A STA $FD    ;3D=XY*V
C39C LDA #$19   ;DIVISIONE
C39E STA $52    ;2D=3D/VZ
C3A0 LDA #$00   ;
C3A2 STA $FC    ;
C3A4 STA $FB    ;
C3A6 STA $FA    ;
C3A8 SEC        ;
C3A9 LDA $FA    ;
C3AB SBC $F7    ;
C3AD TAY        ;
C3AE LDA $FB    ;
C3B0 SBC $F8    ;
```

```
C3B2 TAX            ;
C3B3 LDA $FC        ;
C3B5 SBC $F9        ;
C3B7 BCC $C3BF      ;
C3B9 STA $FC        ;
C3BB STX $FB        ;
C3BD STY $FA        ;
C3BF ROL $FD        ;RISULTATO IN $FD-FE
C3C1 ROL $FE        ;
C3C3 ROL $FF        ;
C3C5 ROL $FA        ;
C3C7 ROL $FB        ;
C3C9 ROL $FC        ;
C3CB DEC $52        ;
C3CD BNE $C3AB      ;
C3CF LDA $50        ;NON COMPLEMENTA SE
C3D1 BPL $C3DA      ;SE FLAG POS/NEG<$7F
C3D3 LDX #$04       ;ALTRIMENTI
C3D5 LDA $FE
C3D7 JSR $C000      ;COMPLEMENTA RISULTATO
C3DA DEC $63        ;FLAG RAGGIO/COORDINATA
C3DC BPL $C3FB      ;SALTA SE HA CALCOLATO
                    ;RAGGIO
C3DE LDA $51        ;FLAG X/Y
C3E0 BNE $C3E6      ;
C3E2 LDA #$A0       ;SOMMA $A0 SE HA
                    ;CALCOLATO X
C3E4 BNE $C3E8      ;
C3E6 LDA #$64       ;SOMMA $64 SE HA
                    ;CALCOLATO Y
C3E8 CLC            ;
C3E9 ADC $FD        ;
C3EB STA $FD        ;
C3ED LDA $FE        ;
C3EF ADC #$00       ;
C3F1 STA $FE        ;
C3F3 BVC $C3FB      ;SE C'E' OVERFLOW
C3F5 DEC $FE        ;RISULTATO=$FFFF
C3F7 LDA #$FF       ;
C3F9 STA $FD        ;
C3FB RTS            ;
```

TABELLA INDIRIZZI

```
C3FC .WORD $C60D (PLOT)
C3FE .WORD $C620 (DRAW)
C400 .WORD $C639 (CIRCLE)
C402 .WORD $C642 (ARC)
C404 .WORD $C263 (CLEAR)
C406 .WORD $C6AA (GRAF)
C408 .WORD $C6CA (TEXT)
C40A .WORD $C6EA (MGRAF)
C40C .WORD $C72A (COLOR)
```

ROUTINE PRINCIPALE DI ARC

```
C427 LDX #$04       ;
C429 LDA $02DF,X ;TRASFERISCE IL VALORE
C42C STA $02A8,X ;DELL'ANGOLO INIZIALE
C42F DEX            ;
C430 BPL $C429      ;
C432 LDA #$E4       ;INIZIALIZZA PUNTATORE
C434 STA $19        ;ANGOLO FINALE
C436 LDA #$02       ;
C438 STA $1A        ;
C43A JMP $C4B8      ;ESEGUE CIRCLE
```

ROUTINE CALCOLO COORDINATE

```
                    ;CALCOLA ORDINATA DI UN
                    ;PUNTO DEL CERCHIO
C43D LDY #$02       ;TRASFERISCE ANG. INIZ.
C43F LDA #$A8       ;IN FAC (AREA DI LAVORO
C441 JSR $BBA2      ;DEL BASIC)
C444 JSR $E26B      ;ESEGUE SENO DI FAC
C447 LDY #$02       ;
C449 LDA #$B7       ;MOLTIPLICA
C44B JSR $BA28      ;RY*FAC
C44E JSR $B1AA      ;CONVERTE FAC DA
                    ;FLOATING IN INTERO
C451 TAX            ;
C452 TYA            ;
C453 CLC            ;SOMMA L'ORDINATA DEL
C454 ADC $02BE      ;CENTRO DEL CERCHIO E
C457 STA $02C8      ;CONSERVA IL RISULTATO
C45A TXA            ;
C45B ADC $02BF      ;
C45E STA $02CC      ;
C461 LDY #$02       ;STESSO PROCEDIMENTO
C463 LDA #$A8       ;PER CALCOLARE ASCISSA
C465 JSR $BBA2      ;DEI PUNTO DEL CERCHIO
C468 JSR $E264      ;ESEGUE COSENO DI FAC
C46B LDY #$02       ;
C46D LDA #$B2       ;
C46F JSR $BA28      ;RX*FAC
C472 JSR $B1AA      ;CONVERTE FAC IN INTERO
C475 TAX            ;
C476 TYA            ;
C477 CLC            ;SOMMA L'ASCISSA DEL
C478 ADC $02BC      ;CENTRO E CONSERVA
C47B STA $02C9      ;IL RISULTATO
C47E TXA            ;
C47F ADC $02BD      ;
C482 STA $02CA      ;
C485 RTS            ;
```

ROUTINE PRINCIPALE DI CIRCLE

```
C486 LDA #$00       ;ANG. INIZ.=0
C488 STA $02A8      ;
C48B STA $02A9      ;
C48E STA $02AA      ;
C491 STA $02AB      ;
C494 STA $02AC      ;
C497 LDA #$E2       ;INIZIALIZZA PUNTATORI
C499 STA $1A        ;
C49B LDA #$E5       ;ALL'ANGOLO FINALE
C49D STA $19        ;
C49F LDA #$7E       ;STEP=PIGRECO/40
C4A1 STA $02AD      ;(RAPPRESENTAZIONE IN
C4A4 LDA #$20       ;FLOATING POINT)
C4A6 STA $02AE      ;
C4A9 LDA #$D9       ;
C4AB STA $02AF      ;
```

```
C4AE LDA #$78    ;
C4B0 STA $02B0   ;
C4B3 LDA #$C5    ;
C4B5 STA $02B1   ;
C4B8 JSR $C015   ;TRASFER. X1 Y1 RX RY
C4BB JSR $C2F4   ;TRASFORMA 3D IN 2D
C4BE LDY $5D     ;
C4C0 LDA $5E     ;
C4C2 JSR $B391   ;RX IN FLOATING POINT
C4C5 LDX #$B2    ;
C4C7 LDY #$02    ;
C4C9 JSR $BBD4   ;CONSERVA RX FLOATING
C4CC LDA $57     ;
C4CE STA $02BC   ;TRASFERISCE X1
C4D1 LDA $58     ;
C4D3 STA $02BD   ;
C4D6 LDY $5F     ;
C4D8 LDA $60     ;
C4DA JSR $B391   ;RY IN FLOATING
C4DD LDX #$B7    ;
C4DF LDY #$02    ;
C4E1 JSR $BBD4   ;CONSERVA RY FLOATING
C4E4 LDA $5B     ;
C4E6 STA $02BE   ;TRASFERISCE Y1
C4E9 LDA $5C     ;
C4EB STA $02BF   ;
C4EE LDA #$00    ;PONE = 0
C4F0 STA $B5     ;FLAG FINE ESECUZIONE
C4F2 JSR $C43D   ;CALCOLA COORD. PUNTO
C4F5 LDA $02C9   ;
C4F8 STA $02C3   ;X1=X2
C4FB LDA $02CA   ;
C4FE STA $02C4   ;
C501 LDA $02CB   ;
C504 STA $02C7   ;Y1=Y2
C507 LDA $02CC   ;
C50A STA $02C8   ;
C50D LDY #$02    ;PONE IN FAC
C50F LDA #$A8    ;VALORE DELL'ANGOLO
C511 JSR $BBA2   ;ATTUALE
C514 LDY #$02    ;
C516 LDA #$AD    ;
C518 JSR $B867   ;FAC=FAC+STEP
C51B LDY #$02    ;
C51D LDX #$A8    ;IL RISULTATO E'
C51F JSR $BBD4   ;IL NUOVO ANG. ATTUALE
C522 LDA $19     ;
C524 LDY $1A     ;
C526 JSR $BC5B   ;COMPARA FAC E ANG.FIN.
C529 CMP #$FF    ;
C52B BEQ $C53B   ;SE FAC >= DI ANG.FIN.
C52D LDA #$01    ;PONE FLAG
C52F STA $B5     ;FINE ESECUZ.=1
C531 LDY #$04    ;
C533 LDA ($19),Y ;L'ANG.FIN. DIVENTA
C535 STA $02A8,Y ;ANG. ATTUALE
C538 DEY         ;
C539 BPL $C533   ;
C53B JSR $C43D   ;CALCOLA COORD. PUNTO
C53E LDX #$09    ;
C540 LDA $02C3,X ;TRASFERISCE
C543 STA $57,X   ;X1 Y1 X2 Y2
C545 DEX         ;
C546 BPL $C540   ;
C548 LDA $5B     ;
C54A STA $4E     ;
C54C LDA $5C     ;
C54E STA $4F     ;
C550 JSR $C55E   ;DISABILITA ROM
C553 JSR $C048   ;ESEGUE ROUTINE DRAW
C556 JSR $C56D   ;RIABILITA ROM
C559 LDA $B5     ;
C55B BEQ $C4F5   ;RIPETE SE FLAG=0
C55D RTS         ;
```

DISABILITA ROM E INTERRUPT

```
C55E LDA #$FE    ;TOGLIE L'INTERRUPUT
C560 AND $DC0E   ;
C563 STA $DC0E   ;
C566 LDA #$FD    ;TOGLIE LA ROM
C568 AND $01     ; DEL KERNAL
C56A STA $01     ;
C56C RTS         ;
```

RIABILITA ROM E INTERRUPT

```
C56D LDA #$02    ;RIMETTE LA ROM
C56F ORA $01     ;
C571 STA $01     ;
C573 LDA #$01    ;RIMETTE L'INTERRUPT
C575 ORA $DC0E   ;
C578 STA $DC0E   ;
C57B RTS         ;
```

CONTROLLA OPZIONE $

```
C57C LDX #$24    ;SE IL CARATTERE
C57E TXA         ;SUCCESSIVO E' $
C57F LDY #$00    ;
C581 CMP ($7A),Y
C583 BEQ $C58B   ;RITORNA,ALTRIMENTI
C585 JSR $AD8A   ;CERCA UN PARAMETRO
C588 LDX #$00    ;NUMERICO
C58A RTS         ;
C58B JSR $0073   ;
C58E RTS         ;
```

PRENDE TRE PARAMETRI

```
C58F JSR $C57C   ;CONTROLLA L'OPZIONE $
C592 CPX #$24    ;SE LA RISCONTRA
C594 BEQ $C59F   ;LASCIA IL VECCHIO VA-
                 ;LORE
C596 JSR $B1AA   ;ALTRIMENTI PRENDE IL
C599 STY $02D0   ;NUOVO VALORE
C59C STA $02D1   ;
C59F JSR $AEFD   ;E CONTROLLA LA VIRGOLA
C5A2 JSR $C57C   ;RIPETE PER IL
C5A5 CPX #$24    ;SECONDO VALORE
C5A7 BEQ $C5B2   ;
C5A9 JSR $B1AA   ;
C5AC STY $02D2   ;
C5AF STA $02D3   ;
C5B2 JSR $AEFD   ;
```

```
C5B5 JSR $C57C    ;E ANCHE PER IL TERZO
C5B8 CPX #$24     ;
C5BA BEQ $C5CB    ;
C5BC JSR $B1AA    ;
C5BF TAX          ;SE IL VALORE DEL PARA-
                  ;METRO Z E' NEGATIVO
C5C0 BPL $C5C5    ;ILLEGAL QUANTITY
C5C2 JMP $B248    ;
C5C5 STY $02D4    ;ALTRIMENTI RITORNA
C5C8 STA $02D5    ;
C5CB RTS          ;
```

PRENDE DUE PARAMETRI

```
C5CC JSR $AEFD    ;CONTROLLA LA VIRGOLA
C5CF JSR $C57C    ;CONTROLLA L'OPZIONE $
C5D2 CPX #$24     ;SE ESISTE LASCIA I
C5D4 BEQ $C5DF    ;VECCHI VALORI
C5D6 JSR $B1AA    ;ALTRIMENTI NE PRENDE
C5D9 STY $02D6    ;DI NUOVI
C5DC STA $02D7    ;
C5DF JSR $AEFD    ;RIPETE PER IL SECONDO
C5E2 JSR $C57C    ;PARAMETRO
C5E5 CPX #$24     ;
C5E7 BEQ $C5F2    ;
C5E9 JSR $B1AA    ;
C5EC STY $02D8    ;
C5EF STA $02D9    ;
C5F2 RTS          ;
```

PRENDE UN PARAMETRO POSITIVO

```
C5F3 JSR $AEFD    ;CONTROLLA LA VIRGOLA
C5F6 JSR $C57C    ;CONTROLLA L'OPZIONE $
C5F9 CPX #$24     ;SE LA RISCONTRA
C5FB BEQ $C60C    ;LASCIA VECCHIO VALORE
C5FD JSR $B1AA    ;ALTRIMENTI PRENDE
C600 TAX          ;IL NUOVO E SE QUESTO
C601 BPL $C606    ;NON E' POSITIVO
C603 JMP $B248    ;ILLEGAL QUANTITY
C606 STY $02DA    ;
C609 STA $02DB    ;


C60C RTS          ;
```

*** PLOT ***

```
C60D JSR $C58F    ;PRENDE 3 PARAMETRI
C610 JSR $C01F    ;SPOSTA I VALORI
C613 JSR $C2EC    ;METTE IN 2D
C616 JSR $C55E    ;TOGLIE LA ROM
C619 JSR $C18D    ;ESEGUE PLOT
C61C JSR $C56D    ;RIMETTE LA ROM
C61F RTS          ;
```

*** DRAW ***

```
C620 JSR $C58F    ;PRENDE 3 PARAMETRI
C623 JSR $C5CC    ;PRENDE 2 PARAMETRI
C626 JSR $C5F3    ;PRENDE 1 PARAMETRO
C629 JSR $C015    ;SPOSTA I VALORI
C62C JSR $C300    ;METTE IN 2D
C62F JSR $C55E    ;TOGLIE LA ROM
C632 JSR $C048    ;ESEGUE DRAW
C635 JSR $C56D    ;RIMETTE LA ROM
C638 RTS          ;
```

*** CIRCLE ***

```
C639 JSR $C58F    ;PRENDE 3 PARAMETRI
C63C JSR $C67E    ;PRENDE I VALORI DEI
                  RAGGI E CONTROLLA CHE
                  SIANO POSITIVI
C63F JMP $C486    ;ESEGUE CIRCLE
```

*** ARC ***

```
C642 JSR $C58F    ;PRENDE 3 PARAMETRI
C645 JSR $C67E    ;PRENDE I VALORI DEI
                  ;RAGGI E CONTROLLA CHE
                  ;SIANO POSITIVI
C648 JSR $AEFD    ;CONTOLLA LA VIRGOLA
C64B JSR $C57C    ;CONTROLLA L'OPZIONE $
C64E CPX #$24     ;SE RISCONTRATA PASSA
C650 BEQ $C659    ;AL PROSSIMO PARAMETRO
C652 LDX #$DF     ;ALTRIMENTI PRENDE
C654 LDY #$02     ;L'ANGOLO INIZIALE E LO
C656 JSR $BBD4    ;PONE A $02DF-$02E3
C659 JSR $AEFD    ;CONTROLLA LA VIRGOLA
C65C JSR $C57C    ;CONTROLLA L'OPZIONE $
C65F CPX #$24     ;SE USATA PASSA AL
C661 BEQ $C66A    ;PROSSIMO PARAMETRO
C663 LDX #$E4     ;ALTRIMENTI PRENDE
C665 LDY #$02     ;L'ANGOLO FINALE E LO
C667 JSR $BBD4    ;PONE A $02E4-$02E7
C66A JSR $AEFD    ;CONTROLLA LA VIRGOLA
C66D JSR $C57C    ;CONTROLLA L'OPZIONE $
C670 CPX #$24     ;SE USATA
                  ;LASCIA I VECCHI VALORI
C672 BEQ $C67B    ;ALTRIMENTI PRENDE
C674 LDX #$AD     ;L'INCREMENTO E LO PONE
C676 LDY #$02     ;A $02AD-$02B0
C678 JSR $BBD4    ;
C67B JMP $C427    ;ESEGUE ARC

C67E JSR $C5CC    ;PRENDE 2 PARAMETRI
C681 LDA $02D7    ;DEVONO ESSERE POSITIVI
C684 BPL $C689    ;
C686 JMP $B248    ;ALTRIMENTI ERRORE
C689 LDA $02D9    ;
C68C BMI $C686    ;
C68E RTS          ;
C68F NOP          ;
C690 NOP          ;
```

ROUTINE VALUTAZIONE COLORI

```
C691 JSR $B79E    ;VALUTA ESPRESS. 0-255
C694 CPX #$10     ;RISULTATO IN X
C696 BMI $C69B    ;SE >$0E
C698 JMP $B248    ;ILLEGAL QUANTITY
C69B STX $FC      ;CONSERVA CODICE COLORE
C69D JSR $AEFD    ;CONTROLLA LA VIRGOLA
C6A0 JSR $B79E    ;IDEM PER PARAMETRO
C6A3 CPX #$10     ;SUCCESSIVO
C6A5 BPL $C698    ;
C6A7 STX $FB      ;
```

```
C6A9 RTS       ;

*** GRAF ***

C6AA JSR $C691 ;VALUTA COLORI
C6AD JSR $C279 ;SETTA I COLORI
C6B0 LDA #$EF  ;
C6B2 AND $D016 ;RESETTA BIT MULTICOLOR
C6B5 STA $D016 ;
C6B8 LDA #$AA  ;CARICA IN A COD. TAX
C6BA STA $C1CF ;LO METTE NELLA ROUTINE
               ;DI PLOT
C6BD LDA #$BD  ;IN A COD. LDA$,X
C6BF STA $C1D0 ;NELLA ROUT. DI PLOT
C6C2 LDA #$E3  ;IN A $E3
C6C4 STA $C1D1 ;NELLA ROUT. DI PLOT
C6C7 JMP $C2A1 ;VA IN HIRES

*** TEXT ***

C6CA JSR $C691 ;VALUTA COLORI
C6CD LDA $FC   ;
C6CF STA $D021 ;SETTA COLORE DI SFONDO
C6D2 LDA #$D8  ;
C6D4 STA $FA   ;INIZIALIZZA PUNTATORI
C6D6 LDY #$00  ;
C6D8 STY $F9   ;A INIZIO MEMORIA COLORE
C6DA LDA $FB   ;
C6DC JSR $C290 ;SETTA COLORE CARATTERI
C6DF LDA #$EF  ;
C6E1 AND $D016 ;RESETTA BIT MULTICOLOR
C6E4 STA $D016 ;
C6E7 JMP $C2B7 ;VA IN MODO TESTO

*** MGRAF ***

C6EA JSR $C691 ;VALUTA PRIMI 2 COLORI
C6ED LDA $FC   ;
C6EF STA $D021 ;SETTA COLORE 0
C6F2 LDA $FB   ;
C6F4 STA $FD   ;CONSERVA 2' COLORE
C6F6 JSR $AEFD ;CONTROLLA LA VIRGOLA
C6F9 JSR $C691 ;VALUTA 3' E 4' COLORE
C6FC LDA #$D8  ;
C6FE STA $FA   ;INIZIALIZZA PUNTATORI
C700 LDY #$00  ;
C702 STY $F9   ;A INIZIO MEMORIA COLORE
C704 LDA $FB   ;
C706 JSR $C290 ;SETTA COLORE 3
C709 LDA $FD   ;
C70B STA $FB   ;
C70D JSR $C279 ;SETTA COLORI 2 E 1
C710 LDA #$10  ;
C712 ORA $D016 ;SETTA BIT MULTICOLOR
C715 STA $D016 ;
C718 LDA #$4C  ;CARICA IN A COD. JMP
C71A STA $C1CF ;LO METTE NELLA PLOT
C71D LDA #$24  ;CARICA IN A $24
C71F STA $C1D0 ;LO METTE NELLA PLOT
C722 LDA #$C2  ;CARICA IN A $C2
C724 STA $C1D1 ;LO METTE NELLA PLOT
C727 JMP $C2A1 ;VA IN MODO GRAFICO

*** COLOR ***

C72A JSR $B79E ;VALUTA 0-255
C72D CPX #$04  ;
C72F BMI $C734 ;SE >$03
C731 JMP $B248 ;ILLEGAL QUANTITY ERROR
C734 STX $02   ;CONSERVA VALORE IN $02
C736 RTS       ;
C737 NOP       ;

ABILITA NUOVE ISTRUZIONI

C738 LDA #$43  ;AGGIORNA PUNTATORI
C73A STA $0308 ;PER RICONOSCIMENTO
C73D LDA #$C7  ;NUOVI COMANDI
C73F STA $0309 ;
C742 RTS       ;
C743 JSR $0073 ;CONTROLLA L'ESISTENZA
C746 CMP #$5F  ;DELLA FRECCETTA
C748 BEQ $C750 ;SE NON E' PRESENTE
C74A JSR $0079 ;ESEGUE LA ROUTINE
C74D JMP $A7E7 ;NORMALE DELL'INTERPRETE
               ;BASIC
C750 LDA #$00  ;AZZERA L'INDICATORE
C752 STA $F9   ;DEL COMANDO DA ESEGUIRE
C754 LDA $7A   ;SALVA I PUNTATORI
C756 STA $FC   ;AL CARATTERE CORRENTE
C758 LDA $7B   ;NELLE LOCAZIONI $FC-$FD
C75A STA $FD   ;
C75C LDY #$00  ;AZZERA IL PUNTATORE
               ;AI NUOVI COMANDI E
C75E BEQ $C774 ;INIZIA IL CONFRONTO
C760 LDA $FC   ;RIPOSIZIONA I
C762 STA $7A   ;PUNTATORI AL CARATTERE
C764 LDA $FD   ;CORRENTE
C766 STA $7B   ;
C768 INC $F9   ;PASSA AL COMANDO
               ;SUCCESSIVO
C76A LDA #$09  ;SE I COMANDI
C76C CMP $F9   ;SONO FINITI
C76E BEQ $C74A ;SYNTAX ERROR
C770 INY       ;AGGIORNA I PUNTATORI
C771 DEX       ;ALLA PAROLA SUCCESSIVA
C772 BNE $C770 ;
C774 LDA $C7AC,Y ;PRENDE IL NUMERO DEI
               ;CARATTERI DI CUI E'
               ;COMPOSTA LA PAROLA
C777 TAX       ;LI SALVA IN X
C778 JSR $0073 ;PRENDE IL CARATTERE
               ;SUCCESSIVO E
C77B STA $FE   ;LO SALVA
C77D INY       ;CONFRONTA CON UN
C77E LDA $C7AC,Y ;CARATTERE DELLA
C781 CMP $FE   ;PAROLA IN ESAME
C783 BNE $C760 ;SE DIVERSO PASSA ALLA
               ;PROSSIMA PAROLA
C785 DEX       ;PASSA AL SUCCESSIVO
               ;CARATTERE DELLA PAROLA
C786 BNE $C778 ;SE QUESTA E' FINITA
C788 ASL $F9   ;PRENDE I PUNTATORI
C78A LDX $F9   ;AL COMANDO
C78C LDA $C3FC,X ;E LI PONE
C78F STA $033F ;A $033F-0340
```

```
C792 INX           ;
C793 LDA $C3FC,X  ;
C796 STA $0340    ;
C799 LDA #$20     ;METTE IL CODICE DI JSR
C79B STA $033E    ;NELLA LOCAZIONE $033E
C79E LDA #$60     ;E QUELLO DI RTS
C7A0 STA $0341    ;NELLA LOCAZIONE $0341
C7A3 JSR $0073    ;PUNTA AL CARATTERE
                  ;SUCCESSIVO
C7A6 JSR $033E    ;ESEGUE IL COMANDO
C7A9 JMP $A7AE    ;PASSA AL PROSSIMO
                  ;COMANDO
```

TABELLA NUOVI COMANDI

```
C7AC $04
C7AD 'PLOT        ;($C60D)
C7B1 $04
C7B2 'DRAW        ;($C620)
C7B6 $06
C7B7 'CIRCLE      ;($C639)
C7BD $03
C7BE 'ARC         ;($C642)
C7C1 $05
C7C2 'CLEAR       ;($C263)
C7C7 $04
C7C8 'GRAF        ;($C6AA)
C7CC $04
C7CD 'TEXT        ;($C6CA)
C7D1 $05
C7D2 'MGRAF       ;($C6EA)
C7D7 $04
C7D8 'COL',$B0    ;($C72A)($B0=CODICE OR)
```

GRSAVE/GRLOAD/GRVERIFY/GRMERGE

ROUTINE DI TRAFERIMENTO

```
C8B4 LDA #$00     ;INIZIALIZZA BYTE BASSO
C8B6 STA $F9      ;DEI PUNTATORI
C8B8 STA $FB      ;
C8BA LDX #$20     ;CONTATORE DI BLOCCHI
C8BC TAY          ;CONTATORE DI BYTES
C8BD JSR $C55E    ;DISABILITA KERNAL
C8C0 JSR $C95B    ;DISABILITA BASIC
C8C3 LDA ($F9),Y ;TRASFERISCE DA $E000
C8C5 BIT $FB      ;
C8C7 STA ($FB),Y ;A $A000 O VICEVERSA
C8C9 INY          ;
C8CA BNE $C8C3    ;RIPETE FINO ALLA FINE
                  ;DEL BLOCCO
C8CC INC $FA      ;PASSA AL PROSSIMO
C8CE INC $FC      ;BLOCCO
C8D0 DEX          ;
C8D1 BNE $C8C3    ;RIPETE FINCHE' NON HA
                  ;TRASFERITO TUTTI I
                  ;BLOCCHI
C8D3 JSR $C56D    ;RIABILITA KERNAL
C8D6 JSR $C960    ;RIABILITA BASIC
C8D9 RTS          ;
```

*** GRSAVE ***

```
C8DA PLA          ;RIMUOVE INDIRIZZO
C8DB PLA          ;DI RITORNO
C8DC LDA #$E0     ;INIZIALIZZA BYTE ALTO
C8DE STA $FA      ;DEI PUNTATORI
C8E0 LDA #$A0     ;
C8E2 STA $FC      ;
C8E4 LDA #$24     ;CARICA IN A COD. BIT$
C8E6 STA $C8C5    ;MODIFICA LA ROUTINE
                  ;DI TRASFERIMENTO
C8E9 JSR $C8B4    ;TRAFERISCE LA PAGINA
                  ;GRAF. DA $E000 A $A000
C8EC JSR $E1D4    ;SET FILE PARAMETERS
C8EF JSR $C95B    ;DISABILITA BASIC
C8F2 LDX #$40     ;IN X/Y ULTIMA LOCAZ.
C8F4 LDY #$BF     ;DA SALVARE
C8F6 LDA #$A0     ;
C8F8 STA $FE      ;IN $FD-FE PRIMA LOCAZ.
C8FA LDA #$00     ;DA SALVARE
C8FC STA $FD      ;
C8FE LDA #$FD     ;IN A INDIRIZZO DEL
                  ;PUNTATORE ALLA PRIMA
                  ;LOCAZ. DA SALVARE
C900 JSR $FFD8    ;SALVA RAM
C903 JSR $C960    ;RIABILITA BASIC
C906 BCC $C90B    ;SE C'E' UN ERRORE
C908 JMP $E0F9    ;ALLORA CONTROLLA DI
                  ;COSA SI TRATTA
C90B RTS          ;
```

*** GRMERGE ***

```
C90C LDA #$11     ;IN A COD. ORA( ),Y
C90E BNE $C912    ;
```

*** GRLOAD ***

```
C910 LDA #$24     ;IN A COD. BIT$
C912 STA $C8C5    ;MODIFICA ROUT. TRASF.
C915 LDA #$E0     ;
C917 STA $FC      ;INIZIALIZZA BYTE ALTO
C919 LDA #$A0     ;DI PUNTATORI
C91B STA $FA      ;
C91D LDA #$00     ;VALORE DI LOAD PER
                  ;FLAG LOAD/VERIFY
C91F BEQ $C923    ;
```

*** GRVERIFY ***

```
C921 LDA #$01     ;VALORE DI VERIFY PER
                  ;FLAG LOAD VERIFY
C923 STA $0A      ;SETTA FLAG LOAD/VERIFY
C925 PLA          ;RIMUOVE INDIRIZZO
C926 PLA          ;DI RITORNO
C927 JSR $E1D4    ;SET FILE PARAMETERS
C92A JSR $C95B    ;DISABILITA BASIC
C92D LDA $0A      ;IN A FLAG LOAD/VERIFY
C92F LDX #$00     ;IN X/Y LOCAZ. INIZIALE
C931 LDY #$A0     ;IN CUI CARICARE O VER.
C933 JSR $FFD5    ;LOAD/VERIFY
C936 JSR $C960    ;RIABILITA BASIC
C939 BCS $C90B    ;SALTA SE C'E' ERRORE
```

```
C93B LDA $0A       ;FLAG LOAD/VERIFY
C93D BEQ $C949     ;SALTA SE ESEGUE LOAD
C93F LDX #$1C      ;IN X COD. VERIFY ERROR
C941 JSR $FFB7     ;LEGGE ST
C944 AND #$10      ;
C946 BNE $C955     ;SALTA SE C'E' ERRORE
C948 RTS           ;
C949 JSR $C8B4     ;TRASF. DA $A000 A $E000
C94C JSR $FFB7     ;LEGGE ST
C94F AND #$BF      ;
C951 BEQ $C948     ;SALTA SE E' TUTTO OK
C953 LDX #$1D      ;ALTRIMENTI CARICA COD.
                   ;DI LOAD ERROR
C955 JMP $A437     ;E SEGNALA L'ERRORE
```

DISABILITA BASIC

```
C958 SEI           ;DISABILITA INTERRUPT
C959 LDA #$FE      ;
C95B AND $01       ;
C95D STA $01       ;DISABILITA BASIC
C95F RTS           ;
```

RIABILITA BASIC

```
C960 PHA           ;CONSERVA A
C961 LDA #$01      ;
C963 ORA $01       ;
C965 STA $01       ;RIABILITA BASIC
C967 CLI           ;RIABILITA INTERRUPT
C968 PLA           ;RIPRENDE A
C969 RTS           ;
```

COMANDI PER SCRIVERE
NELLA PAGINA GRAFICA

*** CHAR ***

```
C970 LDA #$08      ;IN A/X INCREMENTO
C972 LDX #$00      ;DI CHAR
C974 BEQ $C97A     ;
```

*** VCHAR ***

```
C976 LDA #$40      ;IN A/X INCREMENTO
C978 LDX #$01      ;DI VCHAR
C97A STA $B5       ;CONSERVA INCREMENTO
C97C STX $B6       ;
C97E JSR $B79E     ;VALUTA INTERO 0-255
C981 CPX #$28      ;DEVE ESSERE COMPRESO
C983 BCC $C988     ;TRA 0-39 (COLONNA)
C985 JMP $B248     ;ALTRIMENTI ILLEGAL
                   ;QUANTITY ERROR
C988 TXA           ;
C989 LDX #$00      ;
C98B JSR $CA45     ;MOLTIPLICA IL VALORE
                   ;PER 8 E CONSERVA IL
                   ;RISULTATO IN $F7-F8
C98E JSR $B7F1     ;CONTROLLA LA VIRGOLA
                   ;E VALUTA 0-255
C991 CPX #$19      ;DEVE ESSERE COMPRESO
C993 BCS $C985     ;TRA 0-24 (RIGA)
C995 STX $F9       ;
C997 LDA #$18      ;
C999 SEC           ;SOTTRAE 24 DAL VALORE
C99A SBC $F9       ;DELLA RIGA
C99C ASL ,A        ;MOLTIPLICA PER 2
C99D TAY           ;
C99E CLC           ;
C99F LDA $C1EB,Y;PRENDE INDIRIZZO DA
C9A2 ADC $F7       ;TABELLA DELLA MAPPA
C9A4 AND #$F8      ;GRAFICA E AGGIUNGE
C9A6 STA $F7       ;COLONNE*8 SENZA CONSI-
C9A8 LDA $C1EC,Y ;DERARE I 3 BYTES MENO
C9AB ADC $F8       ;SIGNIFICATIVI
C9AD STA $F8       ;
C9AF JSR $B7F1     ;CONTROLLA LA VIRGOLA
                   ;E VALUTA 0-255
C9B2 CPX #$04      ;VALORE TRA 0-3 (SET DI
                   ;CARATTERI)
C9B4 BCS $C985     ;ALTRIMENTI ERRORE
C9B6 STX $FB       ;
C9B8 ASL $FB       ;MOLTIPLICA PER 4
C9BA ASL $FB       ;
C9BC JSR $AEFD     ;CONTROLLA LA VIRGOLA
C9BF JSR $AD9E     ;VALUTA ESPRESSIONE
C9C2 BIT $0D       ;
C9C4 BMI $C9CC     ;SE E' UNA STRINGA SALTA
C9C6 JSR $BDDD     ;CONVERTE DA NUMERO
C9C9 JSR $B487     ;A STRINGA
C9CC JSR $B6A6     ;PUNTATORI DELLA STRIN-
                   ;GA IN $22-23 LUNGHEZZA
                   ;IN A
C9CF TAX           ;
C9D0 BEQ $CA39     ;SE STRINGA NULLA ESCE
C9D2 STA $FC       ;CONSERVA LUNGHEZZA
C9D4 LDY #$00      ;
C9D6 STY $FD       ;AZZERA CONTATORE
C9D8 JSR $CA20     ;INDIVIDUA CARATTERE
C9DB BCC $CA02     ;SE E' UN CARATTERE
                   ;SPECIALE SALTA
C9DD LDX #$02      ;
C9DF JSR $CA45     ;INDIVIDUA LA POSIZIONE
C9E2 LDA #$D0      ;DEL CARATTERE DA STAM-
C9E4 CLC           ;PARE NELLA ROM DEI
C9E5 ADC $FB       ;CARATTERI
C9E7 ADC $FA       ;
C9E9 STA $FA       ;
C9EB SEI           ;DISABILITA INTERRUPT
C9EC LDA #$FB      ;
C9EE AND $01       ;
C9F0 STA $01       ;ABILITA ROM CARATTERI
C9F2 LDY #$07      ;
C9F4 LDA ($F9),Y ;COPIA UN CARATTERE
C9F6 STA ($F7),Y ;DALLA ROM DEI CARAT-
C9F8 DEY           ;TERI ALLA PAGINA GRAF.
C9F9 BPL $C9F4     ;
C9FB LDA #$04      ;DISABILITA LA ROM
C9FD ORA $01       ;DEI CARATTERI
C9FF STA $01       ;
CA01 CLI           ;RIABILITA INTERRUPT
CA02 DEC $FC       ;SE LA STRINGA E' STATA
CA04 BEQ $CA39     ;COMPLETATA ESCE
CA06 CLC           ;
CA07 LDA $B5       ;INDIVIDUA NUOVA LOCAZ.
```

```
CA09 ADC $F7     ;DELLA PAGINA GRAFICA
CA0B STA $F7     ;IN CUI STAMPARE IL
CA0D LDA $B6     ;PROSSIMO CARATTERE
CA0F ADC $F8     ;
CA11 STA $F8     ;
CA13 BCS $CA1F   ;SE LA LOCAZIONE ECCEDE
CA15 CMP #$FF    ;IL LIMITE DELLA
CA17 BNE $C9D8   ;PAGINA GRAFICA ESCE
CA19 LDA $F7     ;
CA1B CMP #$40    ;
CA1D BCC $C9D8   ;
CA1F RTS         ;
```

ROUTINE CARATTERE

```
CA20 LDY $FD       ;
CA22 LDA ($22),Y   ;IN A CARATTERE DELLA
                   ;STRINGA
CA24 INY           ;
CA25 STY $FD       ;AGGIORNA CONTATORE
CA27 TAX           ;
CA28 BMI $CA3A     ;SE COD. DEL CARATTERE
                   ;E' > $7F SALTA
CA2A CMP #$20      ;SE COD. < $20 E' UN
CA2C BCC $CA39     ;CARATTERE NON STAMPA-
                   ;BILE QUINDI SALTA
CA2E CMP #$60      ;
CA30 BCC $CA36     ;
CA32 AND #$DF      ;
CA34 BNE $CA38     ;
CA36 AND #$3F      ;
CA38 SEC           ;
CA39 RTS           ;
CA3A AND #$7F      ;CARATTERI SHIFTATI
CA3C CMP #$7F      ;
CA3E BNE $CA42     ;
CA40 LDA #$5E      ;
CA42 CMP #$20      ;
CA44 RTS           ;
```

ROUTINE MOLTIPL. PER 8

```
CA45 LDY #$00      ;MOLTIPLICA A PER 8
CA47 STY $F8,X     ;IL RISULTATO
CA49 ASL ,A        ;VIENE MESSO NELLE
CA4A ASL ,A        ;LOCAZ. $F7+X $F8+X
CA4B ROL $F8,X     ;
CA4D ASL ,A        ;
CA4E ROL $F8,X     ;
CA50 STA $F7,X     ;
CA52 RTS           ;
```

*** INV ***

```
CA53 JSR $B79E     ;VALUTA PARAM. 0-255
CA56 TXA           ;
CA57 BEQ $CA65     ;SE PARAM.=0 SALTA
CA59 LDA #$51      ;IN A CODICE EOR(),Y
CA5B STA $C1D7     ;MODIFICA ROUT. DI PLOT
CA5E STA $C22D     ;HIRES E MULTICOLOR PER
CA61 STA $C237     ;OTTENERE INVERSIONE
CA64 RTS           ;
CA65 LDA #$11      ;IN A COD. OR(),Y
CA67 STA $C1D7     ;RIMETTE A POSTO
CA6A STA $C237     ;LA ROUTINE DI PLOT
CA6D LDA #$91      ;IN A COD. STA(),Y
CA6F STA $C22D     ;IDEM
CA72 RTS           ;
```

DISASSEMBLATO DELLA ROUTINE DI HARDCOPY PER LA STAMPANTE MPS 802

```
0334 LDA #$00      ;COLLOCA L'INDIRIZZO
0336 STA $FD       ;DELLA PAGINA GRAFICA
0338 LDA #$E0      ;NELLE LOCAZ. $FD-FE
033A STA $FE       ;
033C RTS           ;

033D LDY #$07      ;
033F JSR $036C     ;DISABILITA ROM
0342 LDA ($FD),Y   ;PRENDE 8 BYTES DALLA
0344 STA $037C,Y   ;PAGINA GRAFICA E LI
0347 DEY           ;CONSERVA NELLE
0348 BPL $0342     ;LOCAZ. $037C-0383
034A JSR $0374     ;RIABILITA ROM
034D LDY #$07      ;CONTATORE DI BYTE
034F LDX #$07      ;CONTATORE DI BIT
0351 LSR $037C,X   ;PRENDE IL BIT MENO
0354 ROR ,A        ;SIGNIFICATIVO DI OGNI
0355 DEX           ;BYTE E LO PASSA IN A
0356 BPL $0351     ;
0358 STA $0384,Y   ;CONSERVA RISULTATO
035B DEY           ;RIPETE PER TUTTI E 8
035C BPL $034F     ;I BYTES
035E CLC           ;
035F LDA $FD       ;
0361 ADC #$08      ;AGGIORNA I PUNTATORI
0363 STA $FD       ;AL PROSSIMO GRUPPO
0365 LDA $FE       ;DI 8 BYTES
0367 ADC #$00      ;
0369 STA $FE       ;
036B RTS           ;

036C SEI           ;DISABILITA INTERRUPT
036D LDA #$FC      ;DISABILITA BASIC
036F AND $01       ;E KERNAL
0371 STA $01       ;
0373 RTS           ;

0374 LDA #$03      ;RIABILITA BASIC
0376 ORA $01       ;E KERNAL
0378 STA $01       ;
037A CLI           ;RIABILITA INTERRUPT
037B RTS           ;
```

Disassemblato lente d'ingrandimento con le routines grafiche 3D.

PRIMO BLOCCO :$C806-$C8B0

```
XC806 JSR $C58F    ;prende i parametri
      LDA $02D0    ;e controlla che la x
```

```
        STA $57       ;sia inferiore a 512
        LDA $02D1     ;
        AND #$FE      ;
        BEQ XC81A     ;
XC815   LDX #$0E      ;se e' superiore a 512
        JMP ($0300)   ;ILLEGAL QUANTITY
XC81A   LDA $02D1     ;muove le coordinate
        STA $58       ;nell'area di lavoro
        LDA $02D2     ;e se la Y e' superi-
        STA $59       ;re a 255 scrive
        LDA $02D3     ;il messaggio di
        BNE XC815     ;errore
        LDA $02D4     ;prende il colore
        STA $5B       ;e lo pone in $5b
        LDA $02D5     ;se e' superiore a
        BNE XC815     ;255, errore.
        LDA $CF00     ;Prende il colore
        AND #$0F      ;di fondo della
        STA $5A       ;pagina grafica
        LDA #$2C      ;dispone l'indirizzo
        STA $CFF8     ;di partenza degli
        LDA #$2D      ;sprites 0 e 1
        STA $CFF9     ;
        LDA #$03      ;accende gli sprites
        STA $D015     ;e li allarga su
        STA $D017     ;X e Y
        STA $D01D     ;
        LDA $5A       ;colora lo sprite 0,
        STA $D028     ;della lente e lo
        LDA $5B       ;sprite 1,di copertura
        STA $D027     ;
        SEI           ;toglie l'interrupt
        LDA #$35      ;e il sistema opera-
        STA $01       ;tivo.
        LDA #$00      ;azzera i registri
        STA $5E       ;di lavoro
        STA $02       ;
        JSR XCA73     ;definisce lo sprite
        LDY #$00      ;e calcola
        LDA $57       ;le coordinate
        CLC           ;degli sprite
        ADC #$0C      ;
        BCC XC872     ;
        LDY #$03      ;
XC872   STA $D000     ;
        STA $D002     ;
        LDA $59       ;
        CLC           ;
        ADC #$12      ;
        STA $D001     ;
        STA $D003     ;
        LDA $58       ;e se eccedono 255
        BEQ XC889     ;setta l' MSB
        LDY #$03      ;
XC889   STY $FB       ;
        LDA $D010     ;
        AND #$FC      ;
        ORA $FB       ;
        STA $D010     ;
        LDX #$40      ;trasferisce lo
        LDY #$FF      ;sprite ingrandito
XC899   LDA $03BF,X   ;e riempie lo sprite
        STA XCAFF,X   ;di copertura
        TYA           ;
        STA $CB3F,X   ;
        DEX           ;
        BNE XC899     ;
        LDA #$37      ;rimette il S.O.
        STA $01       ;e l'interrupt
        LDA #$60      ;e il codice di
        STA $0341     ;RTS a $0341
        CLI           ;
        RTS           ;
```

SECONDO BLOCCO:$CA73-$CAFF

```
XCA73   LDA $59       ;prende le Y
        LSR ,A        ;divide per 8
        LSR ,A        ;e moltiplica la
        LSR ,A        ;parte intera per 320
        ASL ,A        ;
        STA $FB       ;
        LDA #$A0      ;
        STA $FC       ;
        JSR XCAE6     ;
        LDA $57       ;prende il byte basso
        AND #$07      ;delle x e usa i bit
        TAX           ;0-2 come puntatore
        LDA $57       ;al dot
        AND #$F8      ;somma i bit 3-7
        STA $5A       ;della x ai bit 0-2
        LDA $59       ;della y
        AND #$07      ;e addiziona il
        ORA $5A       ;risultato al punta-
        CLC           ;tore al byte
        ADC $FE       ;
        BCC XCA9A     ;
        INC $FF       ;
XCA9A   STA $FE       ;
        LDA $FF       ;
        CLC           ;
        ADC $58       ;
        ADC #$E0      ;dispone la partenza
        STA $FF       ;della pagina grafica
        LDY #$00      ;a $E000
XCAA7   LDA ($FE),Y   ;preleva i punti
        STA $005A,Y   ;e li mette nell'area
        LDA $FE       ;di lavoro convertendo
        CLC           ;al formato degli
        ADC #$07      ;sprites.
        BCC XCAB5     ;
        INC $FF       ;
XCAB5   STA $FE       ;
        INY           ;
        CPY #$04      ;ripete per 4 bytes,
        BNE XCAA7     ;
        TXA           ;e usando il punta-
        BEQ XCACA     ;tore al dot
XCABF   ASL $5D       ;esegue la centratura
        ROL $5C       ;nello sprite.
        ROL $5B       ;
        ROL $5A       ;
        DEX           ;
        BNE XCABF     ;
XCACA   LDY $5E       ;usando il puntatore
        LDX #$00      ;sposta il risultato
```

```
XCACE LDA $5A,X     ;a $03c0-03fe
      STA $03C0,Y ;
      INC $5E     ;
      INY         ;ripete per i tre
      INX         ;bytes orizzontali
      CPX #$03    ;
      BNE XCACE   ;
      INC $59     ;
      INC $02     ;
      LDA $02     ;
      CMP #$15    ;e per i 21 bytes
      BNE XCA73   ;vericali,
      RTS         ;poi esce.
```

ROUTINE DI MOLTIPLICAZIONE BINARIA.
moltiplicandi: $FB-$FC
risultato: $FE-$FF

```
XCAE6 LDA #$00    ;dispone la parte
                  ;bassa del risultato
                  ;a zero,e il numero
      LDX #$08    ;di bit a otto.
XCAEA ASL ,A      ;shift a sinistra
      ROL $FB     ;e addiziona se
      BCC XCAF6   ;riscontra 1 nel carry
      CLC         ;
      ADC $FC     ;
      BCC XCAF6   ;
      INC $FB     ;
XCAF6 DEX         ;ripete per otto bit
      BNE XCAEA   ;
      STA $FE     ;e pone il risultato
      LDA $FB     ;in $FE-$FF
      STA $FF     ;
XCAFF RTS         ;
```

Da $CB00 a $CB3E viene contenuto lo sprite della lente
Da $CB40 a $CB7E viene contenuto lo sprite di copertura.

Da $CB80 a $CBFF spazio libero per altre routines LM o per due sprites.

Disassemblato routine Hard/Copy 801/803

Indirizzo partenza: $9f00
Indirizzo fine: $9feb
Start: SYS 40704
Per interrompere: RUN/STOP

```
      JMP X9F89   ;esegue copy

X9F03 JSR X9F1F   ;chiude il file
      LDA #$60    ;apre il canale 96
      LDX #$04    ;sulla stampante
      LDY #$00    ;con ind.secondario 0
      JSR $FFBA   ;
      LDA #$00    ;e nome nullo
      JSR $FFBD   ;
      JSR $FFC0   ;
      LDX #$60    ;predispone per
      JMP $FFC9   ;output ed esce

X9F1C JSR $ABB5   ;predispone le perife-
X9F1F LDA #$60    ;riche al default e
      JMP $FFC3   ;chiude il canale 96
                  ;ed esce

X9F24 LDA #$00    ;cancella il byte di
      STA X9FEC   ;out-screen
      LDA $58     ;se X<0 o X>319
      BEQ X9F3C   ;
      CMP #$01    ;
      BEQ X9F36   ;
X9F31 INC X9FEC   ;setta out-screen a 1
      CLC         ;e predispone punto
      RTS         ;spento

X9F36 LDA $57     ;
      CMP #$40    ;
      BCS X9F31   ;
X9F3C LDA $4F     ;se Y<0 o Y>199
      BNE X9F31   ;esce con punto spento
      LDA $4E     ;e out-screen a 1
      CMP #$C8    ;
      BCS X9F31   ;
      LDA $4E     ;prende i bit 0-2
      AND #$07    ;della Y e li mette
      STA $F7     ;nel risulato
      LDA $4E     ;divide per otto la Y
      LSR ,A      ;e moltiplica per due
      LSR ,A      ;la parte intera
      AND #$FE    ;del risultato
      TAY         ;e lo usa come punta-
      LDA $C1EB,Y;tore alla tabella
      SEC         ;di conversione
      SBC $F7     ;preleva i valori
      STA $F7     ;dalla tabella,
      LDA $C1EC,Y;sottrae loro il ri-
      CLC         ;sultato memorizzato
      ADC $58     ;in precedenza e li
      STA $F8     ;pone nel risultato
      LDA $57     ;dopo avere addizionato
      AND #$F8    ;la parte alta delle X
      TAY         ;usa i bit 0-2 della
      LDA $57     ;parte bassa delle X
      AND #$07    ;come puntatori al dot
      TAX         ;e i bit 3-7 come
      CLC         ;puntatori al byte di
      RTS         ;schermo

X9F6F JSR X9F24   ;controlla i limiti
      JSR $C55E   ;toglie il S.O.
      LDA X9FEC   ;controlla l'out-screen
      BNE X9F82   ;se e' a 1 predispone
      LDA $C1E3,X;punto spento altri-
      AND ($F7),Y;menti controlla sulla
      SEC         ;pagina grafica se il
      BNE X9F83   ;punto e acceso o spen-
X9F82 CLC         ;to, dispone i flag
X9F83 PHP         ;
      JSR $C56D   ;e riprinstina il S.O.
      PLP         ;
```

```
       RTS                 ;
X9F89  JSR X9F03           ;apre il canale su
       LDA #$08            ;stampante e dispone
       JSR $AB47           ;la periferica in modo
       LDA #$C7            ;grafico
       STA $FB             ;mette la Y alla prima
X9F95  LDA #$00            ;riga di schermo
       STA $57             ;e la X alla prima
       STA $58             ;colonna
       STA $4F             ;
X9F9D  LDX #$07            ;dispone il numero di
       STX $FD             ;punti verticali da
       LDA $FB             ;inviare alla stampante
       STA $4E             ;
X9FA5  JSR X9F6F           ;analizza il punto
       ROR $FC             ;tornando con SEC o
       DEC $4E             ;CLC e crea il byte da
       DEC $FD             ;inviare in $FC
       BNE X9FA5           ;decrementa la Y e
       ROR $FC             ;il contatore dei bit
       LDA $FC             ;verticali,terminati i
       ORA #$80            ;punti del byte setta
       JSR $AB47           ;il bit 7 e invia a
       LDA $91             ;se e' premuto
       CMP #$7F            ;RUN/STOP
       BEQ X9FDF           ;esce
       INC $57             ;altrimenti passa
       BNE X9FC5           ;al prossimo punto X
       INC $58             ;
X9FC5  LDA $58             ;e se X e' uguale a
       BEQ X9F9D           ;320
       LDA $57             ;
       CMP #$40            ;
       BNE X9F9D           ;
       LDA #$0D            ;invia un CR alla
       JSR $AB47           ;stampante e
       LDA $FB             ;incrementa le y di
       SEC                 ;sette
       SBC #$07            ;
       STA $FB             ;se non sono terminati
       CMP #$FC            ;i punti in verticale
       BNE X9F95           ;ripete, altrimenti
X9FDF  LDA #$0F            ;mette la stampante
       JSR $FFD2           ;in modo testo,
       LDA #$0D            ;stampa un CR,
       JSR $FFD2           ;
       JMP X9F1C           ;chiude il canale 96
                           ;e torna al basic
X9FEC  .BYTE 0             ;flag di out-screen
END OF ASSEMBLY.
```

ISTRUZIONE		Immediato			Assoluto			Pagina Zero			Accumulatore			Implicito			(IND
Mnemonico	OPERAZIONE	#	$	N	#	$	N	#	$	N	#	$	N	#	$	N	#
ADC	A + M + C → A,C Addiziona con Carry.	105	69	2	109	6D	3	101	65	2							097
AND	A ∧ M → A And logico fra accumulatore e memoria	041	29	2	045	2D	3	037	25	2							033
ASL	C ← 7 6 5 4 3 2 1 0 ← 0				014	OE	3	006	06	2	010	0A	1				
BCC	Branch on Carry Cleared (C = 0) Salta se carry = 0																
BCS	Branch on Carry Set (C = 1) Salta se Carry = 1																
BEQ	Branch on EQual (Z = 1) Salta se uguale																
BIT	A ∧ M (A non viene modificato) And logico fra accumulatore e memoria				044	2C	3	036	24	2							
BMI	Branch on MInus (n = 1) Salta se negativo																
BNE	Branch on Non Equal (z = 0) Salta se non è uguale																
BPL	Branch on Plus (n = 0) Salta se positivo																
BRK	PC + 2 → S, P → S Contatore di programma e status nello stack													000	00	1	
BVC	Branch on oVerflow Cleared (V = 0) Salta se overflow = 0																
BVS	Branch on oVerflow Set (V = 1) Salta se overflow = 1																
CLC	0 → C Pone a 0 il carry													024	18	1	
CLD	0 → D (lavora in esadecimale) Pone a 0 il modo decimale													216	D8	1	
CLI	0 → I Pone a 0 l'interrupt													088	58	1	
CLV	0 → V Pone a 0 l'overflow													184	B8	1	
CMP	A - M Compara l'accumulatore alla memoria	201	C9	2	205	CD	3	197	C5	2							193
CPX	X - M Compara x alla memoria	224	E0	2	236	EC	3	228	E4	2							
CPY	Y - M Compara y alla memoria	192	C0	2	204	CC	3	196	C4	2							
DEC	M - 1 → M Decrementa la memoria				206	CE	3	198	C6	2							
DEX	X - 1 → X Decrementa X							202	CA	2							
DEY	Y - 1 → Y Decrementa Y							136	88	2							
EOR	A ⊻ M → A EOR logico fra accumulatore e memoria	073	49	2	077	4D	3	069	45	2							065
INC	M + 1 → M Incrementa la memoria				238	EE	3	230	E6	2							
INX	X + 1 → X Incrementa X													232	E8	1	
INY	Y + 1 → Y Incrementa Y													200	C8	1	
JMP	JuMP to new location Salta ad una nuova locazione				076	4C	3										
JSR	Jump SubRoutine Salta ad una sottoprocedura				032	20	3										

,X)		(IND),Y			Pag. Zero,x			Assoluto, x			Assoluto, y			Relativo			Indiretto			Pag. Zero, y			STATUS					
$	N	#	$	N	#	$	N	#	$	N	#	$	N	#	$	N	#	$	N	#	$	N	N	Z	C	I	D	V
1	2	113	71	2	117	75	2	125	7D	3	121	79	3										*	*	*	–	–	*
1	2	049	31	2	053	35	2	061	3D	3	057	39	3										*	*	–	–	–	–
					022	16	2	030	1E	3													*	*	*	–	–	–
														144	90	2							–	–	–	–	–	–
														176	BO	2							–	–	–	–	–	–
														240	FO	2							–	–	–	–	–	–
																							M_7	*	–	–	–	M_6
														048	30	2							–	–	–	–	–	–
														208	DO	2							–	–	–	–	–	–
														016	10	2							–	–	–	–	–	–
																							–	–	–	1	–	–
														080	50	2							–	–	–	–	–	–
														112	70	2							–	–	–	–	–	–
																							–	–	0	–	–	–
																							–	–	–	0	–	–
																							–	–	–	–	0	–
																							–	–	–	–	–	0
C1	2	209	D1	2	213	D5	2	221	DD	3	217	D9	3										*	*	*	–	–	–
																							*	*	*	–	–	–
																							*	*	*	–	–	–
					214	D6	2	222	DE	3													*	*	–	–	–	–
																							*	*	–	–	–	–
																							*	*	–	–	–	–
41	2	081	51	2	085	55	2	093	5D	3	089	59	3										*	*	–	–	–	–
					246	F6	2	254	FE	3													*	*	–	–	–	–
																							*	*	–	–	–	–
																							*	*	–	–	–	–
																	108	6C	3				–	–	–	–	–	–
																							–	–	–	–	–	–

		immediato			assoluto			pagina zero			accumulatore			implicito			(in
LDA	M → A Carica l'accumulatore con la memoria	169	A9	2	173	AD	3	165	A5	2							161
LDX	M → X Carica X con la memoria	162	A2	2	174	AE	3	166	A6	2							
LDY	M → Y Carica Y con la memoria	160	A0	2	172	AC	3	164	A4	2							
LSR	0 → 7 6 5 4 3 2 1 0 → C				078	4E	3	070	46	2	074	4A	1				
NOP	No OPeration Nessuna operazione													234	EA	1	
ORA	A V M → A OR logico fra accumulatore e memoria	009	09	2	013	0D	3	005	05	2							001
PHA	A → S Pone l'accumulatore nello stack													072	48	1	
PHP	P → S Pone lo status nello stack													008	08	1	
PLA	S → A Ritira l'accumulatore dallo stack													104	68	1	
PLP	S → P Ritira lo status dallo stack													040	28	1	
ROL	C ← 7 6 5 4 3 2 1 0 ←				046	2E	3	038	26	2	042	2A	1				
ROR	→ 7 6 5 4 3 2 1 0 → C				110	6E	3	102	66	2	106	6A	1				
RTI	ReTurn from Interrupt Ritorna da interrupt													064	40	1	
RTS	ReTurn from Subroutine Ritorna da sottoprocedura													096	60	1	
SBC	A − M − C → A , C Sottrae con carry	233	E9	2	237	ED	3	229	E5	2							225
SEC	1 → C Pone a 1 il carry													056	38	1	
SED	1 → D (esegue in decimale) Pone a 1 il modo decimale													248	F8	1	
SEI	1 → I Pone a 1 l'interrupt													120	78	1	
STA	A → M Pone l'accumulatore nella memoria				141	8D	3	133	85	2							129
STX	X → M Pone x nella memoria				142	8E	3	134	86	2							
STY	Y → M Pone y nella memoria				140	8C	3	132	84	2							
TAX	A → X Trasferisce A in X (a non varia)													170	AA	1	
TAY	A → Y Trasferisce A in Y (a non varia)													168	A8	1	
TSX	S → X Trasferisce lo stack in X													186	BA	1	
TXA	X → A Trasferisce X in A (X non varia)													138	8A	1	
TXS	X → S Trasferisce X nello stack													154	9A	1	
TYA	Y → A Trasferisce Y in A (Y non varia)													152	98	1	

**Codice operativo decimale**
$ **Codice operativo esadecimale**
N **Numero di bytes impegnati**

P **Flag di stato**
S **Stak**
M **Memoria**

LEG

,X)		(ind),Y			page zero,X			Assoluto,X			Assoluto,Y			Relativo			Indiretto			Page zero,Y			
A1	2	177	B1	2	181	B5	2	189	BD	3	185	B9	3										* * – – – –
											190	BE	3							182	B6	2	* * – – – –
					160	B4	2	188	BC	3													* * – – – –
					086	56	2	094	5E	3													0 * * – – –
																							– – – – – –
01	2	017	11	2	021	15	2	029	1D	3	025	19	3										* * – – – –
																							– – – – – –
																							– – – – – –
																							* * – – – –
																							DALLO STACK
					054	36	2	062	3E	3													* * * – – –
					118	76	2	126	7E	3													* * * – – –
																							DALLO STACK
																							– – – – – –
E1	2	241	F1	2	245	F5	2	253	FD	3	249	F9	3										* * * – – *
																							– – 1 – – –
																							– – – – 1 –
																							– – – 1 – –
81	2	145	91	2	149	95	2	157	9D	3	153	99	3										– – – – – –
																				150	96	2	– – – – – –
					148	94	2																– – – – – –
																							* * – – – –
																							* * – – – –
																							* * – – – –
																							* * – – – –
																							– – – – – –
																							* * – – – –

ENDA

∧ **AND**
∨ **OR**
⊻ **OR esclusivo**

* **Modificato**
\- **Non modificato**
M_7 (M_6) Bit 7 (6) della memoria